Technische Universiteit Delft
Faculteit Elektrotechniek, Wiskunde en Informatica
Delft Institute of Applied Mathematics

Het construeren van niet-parametrische
Bayesiaanse netwerken uit data
Constructing non-parametric
Bayesian networks from data

Verslag ten behoeve van het
Delft Institute of Applied Mathematics
als onderdeel ter verkrijging

van de graad van

BACHELOR OF SCIENCE
in de
TECHNISCHE WISKUNDE

door

STEFAN CHRISTIAAN BLAUW

Delft, Nederland
Juli 2014

Copyright © 2014 door Stefan Blauw. Alle rechten voorbehouden.
Constructing non-parametric Bayesian networks from data

S.C. Blauw

Technische Universiteit Delft

Begeleider

Dr. A. M. Hanea

Overige commissieleden

Dr. K.P. Hart
Dr. B. van den Dries

Juli, 2014

Delft
Abstract

Constructing non-parametric Bayesian networks from data
by S.C. Blauw

The objective of this thesis is to design an algorithm for learning the structure of non-parametric Bayesian networks (BNs) from data and has been written as part of acquiring the bachelor’s degree at the Delft Institute of Applied Mathematics of the TU Delft. Inspiration for improving an existing algorithm while designing a new procedure was provided by the study of Sneller in [11].

Non-parametric BNs can be used to model both continuous and hybrid data, but perform best when only continuous data is used. They do not require the severe assumption of the joint normal distribution, and there is no need to discretize the data. Instead, only the normal copula is assumed and has to be validated. This is done by comparing the determinant of the rank correlation matrix of computer generated data under the assumption of the normal copula (DNR) with the determinant of the rank correlation matrix of the actual data (DER).

The BN starts with nodes only, and edges are added based on a high rank correlation (Spearman correlation). Colliders and paths between independent nodes are used to determine the direction of arcs. As this can only be done up to Markov equivalence classes, the last arcs are directed randomly, making sure that no cycles are created.

Based on two public datasets a comparison is made with the well-known PC algorithm. It turns out that the number of edges added by the new algorithm is approximately 1.5 times higher than the number of edges added by the PC algorithm, while some nodes are left unconnected by the new algorithm when they were found connected by the PC algorithm.

The procedure for directing edges on paths between independent nodes and for finding colliders has improved compared with Sneller. Using an artificial dataset that contains independent nodes, the algorithm modeled all but one independence relations well. It was not possible to entirely model the independencies correct, as another independence would have been violated. A few arcs that could not be learned from the data were directed randomly.
Chapter 1

Introduction

This thesis, that has been written as part of acquiring the bachelor’s degree in applied mathematics, describes a way of constructing Bayesian networks (BNs) from data. Bayesian networks, also called Bayesian Belief Networks as they may represent someone’s belief in dependencies among variables, have applications in many (scientific) areas. This chapter describes the advantages and disadvantages of BNs when modeling multivariate distributions, the objective of our research and a summary of the entire thesis.

1.1 Why Bayesian networks?

Bayesian networks have many interesting real world applications and are applied in - for example - the following areas [3]:

– Medical diagnosis
– Troubleshooting (the famous former Microsoft Office assistant ‘paperclip’ was actually built on BNs\(^1\))
– Data mining
– Pattern recognition

Of course the list could go on and on, but an interesting question to answer here is why Bayesian networks are so useful and thus have so many applications. Let us start with a picture (taken from [5]):

Figure 1.1: Example of a model of the direction of a car

Figure 1.1 represents an exemplary model of the direction of a car. Four variables and their mutual relations are shown. \(A, B\) and \(C\) are called the parents of \(D\), and later on it will be shown that \(A\) is dependent of \(B\) when \(D\) is known. This structure is called a collider and plays

\(^1\)Source: http://www.quora.com/Bayesian-Networks/What-are-some-real-life-applications-of-Bayesian-Belief-Networks
a great role in learning BNs from data. Although the given relations seem to be causal, this
need not be the case in general. A nice property is that a directed acyclic graph can be
understood more easily and intuitively by people than more abstract statistical concepts such
as cumulative distribution functions. Margaritis analyzes in [5] what makes BNs so useful:

- Graphical models can be understood clearly and intuitively.
- As they are directed graphs, causal relationships can be modeled.
- Uncertainty can be taken into account using probabilistic theories.
- Both direct and indirect relationships can be represented.

Of course there are also disadvantages. The process can become quite unautomated when
directions of edges cannot be determined and experts have to assess the BNs produced. People
also tend to add edges to the graph based on their knowledge, which does not only make the
graph more complicated, but also requires more data for a reliable quantification.

When learning a BN from data, a broad range of mathematical concepts plays a role. These
are described in chapter 2. Correlation is the basis for adding edges to the network. The other
- more complicated - part is directing the edges in such a way that the (conditional)
independencies found in a dataset are not violated and the graph remains acyclic. Doing this
in a proper way is the objective of our research.

1.2 Objective

In December 2013, M. Sneller published her bachelor’s thesis at the Delft Institute of Applied
Mathematics on implementing quantitative expert judgment in learning the structure of
non-parametric Bayesian networks in [11]. In her recommendations, she mentioned some
problems she had encountered, including the following two:

1. No colliders were found in the data she had used, but it remained unclear why.
2. In order to preserve independence between two independent nodes A and B, she decided
to direct all edges adjacent to A and B in a path between them outwards (not pointing
to A or B). This however is not required and is thus an undesired limitation.

Other recommendations regarded the implementation of expert judgment, which is not part of
this thesis and will not be taken into account. The objective of this thesis is to design a new
algorithm for learning non-parametric BNs from data, trying to improve on the problems that
Sneller encountered during her research. Several parts have been changed or added to the
algorithm, and the MATLAB code has been built from scratch, making it an entirely new
algorithm in fact.

1.3 Contents of thesis

In chapter 2, the mathematical principles that form the basis for BNs are described. The
definition of BNs is given, and concepts like (conditional) independence, correlation and
copulas are described. In chapter 3, the algorithm used to learn the structure from datasets is
described in detail, including the independence test developed by Zhang. Chapter 4 presents
an overview of the results and a comparison with the results of other algorithms that have
been applied to these datasets. Finally, in chapter 5 our conclusions are presented, together
with a discussion on them. In the appendix all details on the datasets used and the
MATLAB-code that was implemented can be found.
Chapter 2

Mathematical principles

Before we can start our actual research and design a new algorithm, a sound knowledge about the relevant mathematical ideas and concepts must be gained. This chapter describes the most important mathematical ideas, starting with a detailed explanation of BNs. Thereafter several other principles are introduced, such as correlation, d-separation and copulas.

2.1 Bayesian networks

As this research is concerned with BNs, this gives rise to a number of natural questions we can ask ourselves: what exactly characterizes a BN, and what are its mathematical foundations? This section provides answers to these questions.

2.1.1 Definition

Formally, a Bayesian network consists of two parts [9]:

- A graph consisting of a set of vertices and a set of arcs. This is the graphical part of the BN, which is also called the qualitative part.
- The marginal distributions per node, conditioned on its parental nodes. These can be taken from the data directly or they may be elicited from experts. This is the quantitative part of the BN.

We restrict the qualitative part to be a directed, acyclic graph, abbreviated dag. We can represent the dag by its adjacency matrix $G$ and we say that $G(A, B) = 1 \iff A \rightarrow B$ is a directed edge in the graph. A directed edge is called an arc. Acyclic means that there are no cycles present in the graph. Thus, once we have ‘left’ a certain node $C$, it is impossible to return to it following the directed edges.

One of the strengths of a Bayesian network lies in the possibility of ordering of the nodes: the parents of a node $A$ are defined as $\text{Pa}(A) = \{ B : G(B, A) = 1 \}$. In the same way the children of a node are the nodes that can be reached directly: $\text{Ch}(A) = \{ B : G(A, B) = 1 \}$. Staying with familial terms, the ancestors of a node are the set of all parents, grand-parents etc., that lead to a node. That is, $\text{Anc}(A) = \{ B : B \rightarrow \ldots \rightarrow A \}$. The descendants of a node are finally defined as the set of children, grand-children etc. Thus, $\text{Desc}(A) = \{ B : A \rightarrow \ldots \rightarrow B \}$. This notation is entirely due to [7].

2Some authors use ‘edge’ and ‘arc’ interchangeable, but we make a strict distinction
2.1.2 Dependencies

Conditional dependence is an important characteristic of multivariate distributions. It can be defined as in [9]:

**Definition 1.** Let \(X, Y\) and \(Z\) be random variables and let \(f_X(x), f_Y(y)\) and \(f_Z(z)\) denote their respective density or mass functions, \(f_{X|Y,Z}(x|y,z)\) the density function of \(X\) given \(Y\) and \(Z\), \(f_{X|Z}(x|z)\) the density of \(X\) given \(Z\) and \(f_{XY|Z}(xy|z)\) the joint density of \(X\) and \(Y\) given \(Z\). \(X\) and \(Y\) are conditional independent given \(Z\), and we write, \(X \perp Y \mid Z\) if and only if \(f_{X|Y,Z}(x|y,z) = f_{X|Z}(x|z)\), which is equivalent to \(f_{XY|Z}(xy|z) = f_X|Z(x|z)f_Y|Z(y|z)\).

Thus, conditional independence can be explained in two ways: on the one hand we can say that when the value of \(Z\) is known, knowing the value of \(Y\) does not provide us any additional information regarding \(X\). On the other hand the distribution of \(X\) and \(Y\) given \(Z\) can be factored out as a product.

Each node in the graph is the representation of one random variable (in our case always originating from a data set). The absence of an edge implies (conditional) independence. A node is independent of its ancestors when its parents are known: \(A \perp \text{Anc}(A) \mid \text{Pa}(A)\). The negation of this statement is true as well: (conditional) dependence of variable \(A\) and \(B\) implies the presence of a directed edge between them.

It is generally known that joint density functions can be factored out in terms of conditional distribution functions:

\[
 f_{X_1,\ldots,X_n}(x_1,\ldots,x_n) = \prod_{i=1}^{n} f_{X_i|X_{i+1},\ldots,X_n}(x_i|x_{i+1},\ldots,x_n) \quad (2.1)
\]

The conditioning set in equation 2.1 can become quite large, which causes the computational time to increase when working with BNs. However, if the joint density can be represented by a BN, where a node is independent of its ancestors given its parents, then the previous equation can be simplified to

\[
 f_{X_1,\ldots,X_n}(x_1,\ldots,x_n) = \prod_{i=1}^{n} f_{X_i|\text{Pa}(X_i)}(x_i|\text{Pa}(x_i)) \quad (2.2)
\]

This indeed is the equation that is used to perform calculations involving the cumulative distribution when working with Bayesian networks.

The result of this equation is that BNs can (and are) used to represent probability density functions in a more compact way. This actually results in an exponential reduction of the data, as specifying the pdfs in the standard way requires \(O(2^n)\) parameters, whereas using the compact probabilities \(P(X_i|\text{Pa}(X_i))\) only requires \(O(2^{|	ext{Pa}|+1})\) parameters per node [5].

2.2 Structure

The structure of the BN comprises the arcs that are added and the directions that are given to them. In our algorithm, arcs are added when there is a high rank correlation between nodes, which will be discussed in section 2.3. Directing edges is done based on dependencies found among the variables. We will start with elementary situations, and then introduce the concept of d-separation to be able to deal with more complex structures.
2.2.1 Simple structures

We will start with an elementary situation in which only three nodes are involved. Assume the edges between the nodes have already been added. Let $V_1, V_2$ and $V_3$ be random variables.

![Figure 2.1: Undirected network](image1)

The network depicted in figure 2.1 can be directed in four ways:

![Figure 2.2: Directed network](image2)

The upper left subgraph is called a fork, where $V_2$ is the common cause of $V_1$ and $V_3$. Thus, $V_1$ and $V_3$ are dependent, but become independent when the value of $V_2$ is known. The upper right and lower left structures are called a chain. In this situation we have a sequence where $V_2$ is the middle node and $V_1$ and $V_3$ are the first or last node of the sequence. Obviously, $V_1$ and $V_3$ are again dependent through $V_2$, but become independent when $V_2$ is known. In the lower right situation however, $V_1$ and $V_3$ have a common effect. This v-structure is called a collider and represents that $V_1$ and $V_3$ are independent, but become dependent given $V_2$.

A collider is an important structure, because the relationships $(V_1 \not\perp V_3) \land (V_1 \perp V_3 \mid V_2)$ can be represented by three directed graphs, whereas the dependencies $(V_1 \perp V_3) \land (V_1 \not\perp V_3 \mid V_2)$ can only be represented by a collider. Therefore, finding colliders is a great aid in creating directed graphs.

A skeleton of any dag is the graph that one obtains by replacing all its arcs by edges. A v-structure is a collider. The following equivalence can now be established [1]:

**Theorem 2.2.1 (Markov equivalence).** Two structures are Markov equivalent if and only if they have the same undirected skeleton and the same set of v-structures.
The set of Markov equivalence BNs form what is called a Markov equivalence class [7]. For its formal definition, we refer to [7].

The important implication of Markov equivalence is that we are not able to learn the directions of all arcs in a graph, even when there is an infinite amount of data. The structure can be learnt up to Markov equivalence. A Markov equivalence class can be represented by a partially directed acyclic graph (pdag). The arcs that are undirected in this pdag can be directed in either direction, as long as no cycles are created. The edges that may not be reversed are called compelled edges.

2.2.2 Complex structures

Adding directions based on colliders alone however does not suffice. Besides, there are much more complicated structures possible where more nodes and edges are involved. To be able to deal with these situations, we can make use of the concept of d-separation.

D-separation

D-separation, short for dependence separation, can be defined as in [9]:

Definition 2. Let A, B and C be three disjunct set of nodes. A is said to be d-separated from B by C if and only if

- The path between A and B contains a chain $i \rightarrow m \rightarrow j$ or a fork $i \leftarrow m \rightarrow j$ such that the middle node $m \in C$, or
- The path between A and B contains a collider $i \rightarrow m \leftarrow j$ such that $m \notin C$ and no descendant of $m$ is in C ($\text{Desc}(m) \notin C$).

When two nodes X and Y are d-separated by node Z, then $X \perp Y \mid Z$ in the graph. When X and Y are not d-separated by Z, they are called d-connected. In this case we cannot guarantee the conditional independence.

The definition of d-separation can be explained more intuitively. Assume $Z = \emptyset$ (no third variable is taken into account) and there are $n$ paths that contain at least two edges between X and Y. Then $X \perp Y$ when all paths between X and Y contain a collider (which implies there are $m \geq n$ colliders between X and Y). X and Y are dependent when there is at least one path without a collider (there are $0 \leq m < n$ paths with a collider).

When $Z \neq \emptyset$, this principle works the other way around. Given the observed node(s) in Z, X and Y are d-separated by Z when the elements of Z are the middle nodes of a chain or fork in the paths between X and Y. They are d-connected when the elements of Z are the middle nodes of colliders. Again, all paths have to be ‘blocked’ to hold independence, whereas one ‘open’ path already gives dependence.

Regarding the algorithm, d-separation provides us with rules to direct edges on paths between independent nodes. One choice could be to direct all edges adjacent to independent nodes outwards. This is the most elementary situation, but it is only required if there is one node between X and Y ($X \perp Y$). Therefore, we choose not to do this, but instead perform multiple checks to determine what directions have to be given. Details can be found in chapter 3.
2.3 Correlation

In our algorithm, arcs are added based on a high correlation between nodes. However, one can distinguish between multiple types of correlation. Here we would like to distinguish between linear correlation, also known as Pearson correlation or product-moment correlation, and monotone correlation, also known as rank correlation or Spearman correlation. Partial correlation plays a role in the quantification of non-parametric BNs.

2.3.1 Linear correlation: Pearson

The linear correlation \( \rho \in [-1, 1] \), also called product-moment correlation, between two variables \( V_1 \) and \( V_2 \) can be calculated by

\[
\rho = \frac{\text{Cov}(V_1, V_2)}{\sigma_{V_1} \sigma_{V_2}} = \frac{E(V_1 V_2) - E(V_1)E(V_2)}{\sigma_{V_1} \sigma_{V_2}}
\]

where \( \sigma_{V_1} \) and \( \sigma_{V_2} \) are the standard deviations of \( V_1 \) and \( V_2 \) respectively, \( \text{Cov}(V_1, V_2) \) is the covariance between \( V_1 \) and \( V_2 \) and \( E(V_1) \) and \( E(V_2) \) denote the expected values of \( V_1 \) and \( V_2 \).

A perfect positive linear relationship results in \( \rho = 1 \), whereas a perfect negative linear relationship results in \( \rho = -1 \). When using data, a linear relationship often is too restrictive as it will not be found in the data. A less restrictive alternative is Spearman correlation, which will be introduced after partial correlation.

2.3.2 Partial correlation

Partial correlation is used in the independence test of our algorithm and plays a role when quantifying non-parametric BNs.

Partial correlation of two variables \( V_1 \) and \( V_2 \) with respect to a set of variables \( S \) can be defined as in [4]:

**Definition 3.** The partial correlation of two variables \( V_1 \) and \( V_2 \) given all other variables is

\[
\rho_{V_1V_2;V_3,...V_n} = \frac{K_{12}}{\sqrt{K_{11} K_{22}}}
\]

where \( K_{ij} \) denotes the \((i,j)\)-th cofactor of the correlation matrix.

The application of partial correlation is given in [14]: assume that \( V_1 \), \( V_2 \) and \( V_3 \) are random variables and with a joint normal distribution. Then

\[
\rho_{V_1V_2;V_3} \Rightarrow V_1 \perp V_2 | V_3
\]

because under the joint normal distribution \( \rho_{V_1V_2;V_3} = \rho_{V_1V_2|V_3} \) and \( \rho_{V_1V_2|V_3} \Rightarrow V_1 \perp V_2 | V_3 \). The usefulness of equation 2.3 lies in the fact that calculating partial correlation can be done more easily than conditional correlation. In section 2.5 it is described that conditional rank correlations are used to quantify non-parametric BNs. Here it is important to know that under the assumption of the joint the normal distribution the relation between \( \rho(V_1, V_2) \) and \( r(V_1, V_2) \) is given by:

\[
\rho(V_1, V_2) = 2 \sin\left(\frac{\pi}{6} r(V_1, V_2)\right)
\]

where \( r(V_1, V_2) \) denotes the rank correlation between \( V_1 \) and \( V_2 \) [11].
2.3.3 Rank correlation: Spearman

For a better readable text and to reduce the number of subscripts, the random variables will now be denoted by $X$ and $Y$. The Spearman correlation, which we will denote by $r(X, Y)$, gives the linear correlation between the ranks of the variables, instead of the linear correlation of the actual values. Ranking variables means ordering them from the lowest to the highest value. For example, the observed values $(8, 3, 12)$ would become $(2, 1, 3)$ after ranking. As the ranks of $X$ can be compared to $F_X(X)$, we can write $r(X, Y) = \rho(F_X(X), F_Y(Y))$, and thus $r \in [-1, 1]$. The advantage of Spearman’s correlation is its ability to detect a more general form of correlation: when two variables are - for example - related as $Y = X^3$, then $r(X, Y) = 1$ whereas $\rho(X, Y)$ is about $0.83$, see figure 2.3.

![Figure 2.3: X and Y are monotonically related. $r(X, Y) = 1$, but $\rho(X, Y) = 0.83$.](image)

It can be proven that $F_X(X)$ has a uniform distribution on the unit interval $[0, 1]$ for any given random variable $X$ with a continuous invertible distribution function $F_X$. Let $F_X$ be such a function.

Proof.

\[
P(F_X(X) \leq x) = P(F_X^{-1}(F_X(X)) \leq F_X^{-1}(x))
= P(X \leq F_X^{-1}(x))
= F_X(F_X^{-1}(x))
= x \quad \square
\]

$F_X(X)$ and $F_Y(Y)$ are useful to construct copulas, which will be introduced in the next section and are used in non-parametric BNs.

2.4 Copulas

We would like to define copulas as Nelsen in [8]. This is not the most formal definition there is, but this one is based on its applications in statistics.

Definition 4. A copula of two random variables $X$ and $Y$ is the joint cumulative distribution function of $F_X(X)$ and $F_Y(Y)$ on the unit square $[0, 1]^2$ with uniform marginal distributions.

The usefulness of copulas lies in the fact that a copula together with the marginal distribution of $X$ and $Y$ uniquely defines the joint distribution of $X$ and $Y$, which is known as Sklar’s theorem [8]:

\[
p(X, Y) = C(F_X(X), F_Y(Y))
\]
Theorem 2.4.1 (Sklar’s theorem). For any multivariate distribution \( F(X_1, X_2) \) of real valued variables, there exists a copula \( C \) such that \( F(X_1, X_2) = C(F(X_1), F(X_2)) \). If \( X_1 \) and \( X_2 \) are continuous variables, then \( C \) is unique.

In other words, a copula separates the dependence structure from the marginal distributions. There are many kinds of copulas. An important one is the normal copula:

Definition 5. Let \( R \) be a correlation matrix. Then

\[
C(F(X_1), (X_2)) = \Phi(\Phi^{-1}(F(X_1)), \Phi^{-1}(F(X_2)))
\]

is a normal copula over \([0,1]^2\), where \( \Phi \) is the joint cumulative distribution of a multivariate normal distribution with means zero and covariance matrix \( R \), and \( \Phi^{-1} \) is the inverse standard normal cumulative distribution.

It is important to note that the normal copula alone does not guarantee the joint normal distribution. Having non-normal marginal distributions will lead to a joint distribution unequal to the normal one.

2.5 Non-parametric Bayesian networks

As mentioned before, one can distinguish between discrete, continuous and hybrid BNs. Discrete BNs can be used to model discrete data, or continuous variables may be discretized [2]. This will not be done in this algorithm, as discretizing causes loss of information. Discrete BNs are quantified by conditional probability tables, as illustrated in figure 2.4. This figure is taken from [5] and shows the probability of the 5 values of ‘car direction’ (-60/-30/0/30/60) conditioned on ‘gas pedal pressed’, ‘gear’ and ‘steering wheel angle’.

In case of the continuous BN, it is possible to create a fully-parametric BN, but this usually has the restriction that the joint normal distribution is assumed. This restriction will not be imposed either.
Using our algorithm, a non-parametric BN will be built, as suggested in [2]. The only underlying assumption is that the data follows the normal copula, not the joint the normal distribution. This is a less severe assumption. In order to have a joint normal distribution, both the normal copula and normal marginals are required.

Non-parametric BNs are quantified by their marginal distributions, in our case directly taken from the data, and the conditional rank correlations. This quantification is illustrated in the following figure, which has the same structure as the car network. Now assume $A$, $B$ and $C$ are continuous variables.

![Figure 2.5: Required quantification objects for a continuous BN.](image)

In order to fully quantify this BN, each marginal distribution $f_i$ and all conditional rank correlations are required. Together with the qualitative part, the joint distribution of all variables is uniquely determined.

### 2.6 Measures of fit for the normal copula

When building a non-parametric BN there are two properties that must be validated: before adding edges it must be validated that the data has a normal copula and at the end of the process, when the edges and directions have been determined, it must be validated whether the BN represents enough dependence. One way of doing this is by comparing determinants, of which we use three variants:

- DER: the determinant of the empirical rank correlation matrix.
- DNR: the determinant of the empirical rank correlation matrix under the assumption of the normal copula.
- DBBN: the determinant of the Bayesian network.

In [2] it is discussed why the determinant is a good measure of multivariate dependence. A nice property is that for any given determinant $D$ of a rank correlation matrix, we have $0 \leq D \leq 1$ where $D = 1$ represents complete mutual independence among the variables, while a lower value represents more dependence. In [2] the following theorem is proven:

**Theorem 2.6.1** (Determinant factorization). Let $D$ be the determinant of the $n$-dimensional linear correlation matrix. Then

$$D = \prod_{\text{arcs}} (1 - \rho_{ij}^2; D_{ij})$$  \hfill (2.5)

where $\rho_{ij}^2; D_{ij}$ is the partial correlation associated with the arc between node $i$ and $j$, conditioned on set $D_{ij}$ and the product is taken over all arcs in the BN.
Under the assumption of the normal copula $\rho_{ij:D_{ij}} = \rho_{ij|D_{ij}} \approx r_{ij|D_{ij}}$ [2]. Now, let $R$ be the determinant of the $n$-dimensional rank correlation matrix. Then

$$R \approx \prod_{arcs} (1 - r^{2}_{ij|D_{ij}}). \quad (2.6)$$

A more detailed description of these determinants and the validation process is given below.

**DER**: the empirical rank correlation matrix (ERCM) consists of all pairwise Spearman rank correlations: $\text{ERCM}(i,j) = r(i,j)$. The determinant of this matrix is abbreviated DER: $\text{DER} = \det(\text{ERCM})$.

**DNR**: the DNR is the determinant of the empirical rank correlation matrix under the assumption of the normal copula (ENRCM). In order to calculate this determinant, first the marginal distributions have to be transformed to normal marginals. The marginal distributions are first transformed to the uniform $F_{X_{i}}(X_{i})$ distributions, the ranks. They can be transformed to normal distributions by applying the inverse standard normal distribution $\phi^{-1}$:

$$\tilde{X}_{i} = \phi^{-1}(F_{X_{i}}(X_{i})) \sim N(0, 1)$$

where $X_{i}$ denotes a random variable. Recall that normal marginals together with the normal copula give the joint normal distribution. Also recall that

$$\rho(X_{i}, X_{j}) = \frac{\text{Cov}(X_{i}, X_{j})}{\sigma_{X_{i}} \sigma_{X_{j}}}.$$ 

After transformation $\tilde{X}_{i} \sim N(0, 1)$ and $\tilde{X}_{j} \sim N(0, 1)$, and thus

$$\sigma_{\tilde{X}_{i}} = \sigma_{\tilde{X}_{j}} = 1 \Leftrightarrow \rho(\tilde{X}_{i}, \tilde{X}_{j}) = \text{Cov}(\tilde{X}_{i}, \tilde{X}_{j}) \forall i \neq j.$$ 

Therefore, the transformed variables have a multivariate joint normal distribution with means 0 and the covariance matrix equal to the correlation matrix. By sampling from this distribution and calculating the empirical rank correlation of this sample, one gets the empirical normal rank correlation matrix. The determinant of this matrix is called DNR. If the DER is in the 90% confidence interval of the DNR, then the assumption of the normal copula cannot be rejected.

**DBBN**: the determinant of the rank correlation matrix of a BN that has been constructed on the assumption of the normal copula (which we call here EBRCM) is called the DBBN. The entries of the EBRCM equal the conditional rank correlations that quantify the BN. In a saturated graph $\text{EBRCM} = \text{ENRCM}$.

The DBBN is used to check whether the BN represents enough dependence. Starting with nodes only, $\text{DBBN} = 1$, as the nodes in the graph are independent without edges. When arcs are being added, DBBN starts to decrease towards 0. When the DNR lies in the 90% confidence interval of the DBBN, it can be concluded that enough dependencies have been modeled, although most probably there will still be more dependencies in the data. [2]

Concluding, of the three determinants that are used as a measure, the DNR and DBBN are calculated by assuming the normal copula, while the DER is based on the original data only. To validate whether the normal copula may be assumed and the BN represents enough dependence, it is verified that the DER lies in the 90% confidence interval of the DNR and the DNR lies in the 90% confidence interval of the DBBN, respectively.
Chapter 3

Learning the structure from data

This chapter describes all aspects related to the design of a new structure learning algorithm that is based on the mathematical ideas of chapter 2. First the different kinds of structure learning methods are described, followed by the independence test. Finally, a complete overview of the new algorithm is presented.

3.1 Structure learning methods

There exist many types of learning algorithms for BNs, specialized on a certain type of data. Each type is capable of learning the structure from either discrete, continuous or hybrid data. It has already been decided that a non-parametric BN for continuous variables will be the output of our algorithm. Therefore we now focus on the learning methods itself.

There are two kinds of structure learning methods: score based methods and constraint based methods [5]. The former is based on a scoring that indicates how well the BN fits the data (goal of the algorithm is to maximize this score), whereas the latter is based on independence relations. The algorithm that is concerned in this thesis obviously belongs to the second category, as we try to construct our BN based on the independence relations.

Two well-known constraint based learning algorithms are the PC and IC algorithm. PC is named after its authors Peters and Clark, IC is short for Inductive Causation. One of the biggest disadvantages of these algorithms is that their output is highly influenced by the independence test. Changing the independence test can result in having completely different BNs, which is illustrated in chapter 4. This is the main reason why having a good independence test is of great importance.

The IC algorithm starts with an empty graph (nodes only) and adds edges based on an independence test. After this test is completed, the edges are directed by using colliders and four rules that have been introduced by Verma and Pearl. Aim of these rules is preventing the construction of new colliders and maintaining an acyclic graph. Working the other way around, the input of the PC algorithm is a saturated graph (all edges are present) whose edges are removed using the principle of d-separation [12]. Two rules of our algorithm are based on both the IC and PC algorithm.

3.2 Independence tests

In order to be able to find colliders, an unconditional independence test, as well as a conditional independence test, that perform well are required. However, testing for conditional
independence is not an easy task and traditional methods either focus on the discrete case or make simplifying assumptions [14]. Most times the variables are assumed to have a linear relationship or a joint normal distribution. As these assumptions can fail, they may lead to false conclusions. In case the assumption of the joint normal distribution does hold, testing for conditional independence reduces to testing for zero partial correlation or zero conditional correlation, which can be performed easier.

Zhang explains why new methods that have been introduced recently can be divided into four categories, and what their respective disadvantages are. Avoiding these disadvantages, Zhang introduces a new method for testing conditional independence, which is based on so-called kernel matrices belonging to the three variables $X$, $Y$ and $Z$. A Kernel-based Conditional Independence test - the KCI-test - is the result of his paper. Normal (unconditional) independence is included as a special case of conditional independence.

One way to assess conditional independence is through the estimation of probability densities. As the number of data points required for estimating these probability densities is exponential in the data dimension, this can be a lengthy process. Therefore, Zhang bases the KCI-test on the concept of the cross-covariance operator as a characterization of conditional independence.

In Theorem 4 of his paper, Zhang introduces a known test statistic $T_{UI}$ which has the same asymptotic distribution as another empirical test statistic $\tilde{T}_{UI}$ under the null hypothesis that $X$ and $Y$ are independent. The test is based on validating whether this holds, but for the details of this test we refer to [14] as the level of functional analysis involved is too high to be explained in a bachelor’s thesis.

Regarding performance, Zhang concludes that this method involves less random errors than the methods based on estimating probability densities and uses more information from the data than the methods that are based on discretization. After testing the KCI-test on multiple datasets, Zhang draws the conclusion that this test has a greater accuracy and speed than existing ones. Therefore we feel that the KCI-test is a suitable test for our algorithm.

3.3 The learning algorithm

In this section our own algorithm is described. Thereafter a short explanation on datasets that we use is given, and the way the algorithm has been implemented in MATLAB is described.

3.3.1 Design

Four phases of our algorithm can be distinguished, that will be elaborated on in detail.

1. Validation phase: the assumptions of the BN are validated.
2. Initial phase: an empty graph (nodes only) is filled with $N$ edges\(^3\).
3. Main phase: each edge is directed and becomes an arc.
4. Final phase: validation that the BN represents enough dependence.

Validation phase: This phase has already been described in section 2.6 and is completely performed using Uninet that has an in-built function for these validation steps. More details are given in section 4.2 as well.

\(^3\)This number is actually determined after directing the edges.
Initial phase: Having completed the validation phase, the initial phase of the algorithm is run in MATLAB. Using the KCI-test, all pairs of independent nodes are determined. As the rank correlation matrix has already been computed in the initial phase, edges are added based on a high correlation, in decreasing order. Because independencies may never be violated, it of great importance to never add an arc between independent nodes. The number $N$ of edges that have to be added in total is not known beforehand, but is determined in the final phase. Initially, one has to decide manually on the number of edges to be added. In this paper it was chosen to start with the number of edges as were present in the BNs a comparison is made with. If no comparable BNs are available, one could for example choose the number of nodes as a starting point for the number of edges.

Main phase: The goal of the main phase is to direct as many edges as possible by using different rules that are mostly inspired on the concept of d-separation. The first step is to use colliders, as they determine the direction of two edges uniquely. Hereafter the paths between independent nodes and their lengths are taken into account.

If $V_1 \perp V_3$ and an undirected path between $V_1$ and $V_3$ consists of three nodes ($V_1 - V_2 - V_3$), a collider $V_1 \rightarrow V_2 \leftarrow V_3$ must be created to preserve independence. If the path between $V_1$ and $V_3$ consists of more than three nodes, there are multiple options. One possible choice is to direct the edges from $V_1$ and $V_3$ in the path outwards, which guarantees the presence of a collider somewhere in the path, and thus d-separation gives us independence. However, doing so greatly reduces the amount of possible BNs. As we do not want to impose this restriction, multiple rules have been defined:

(a) If there already is a collider in the path between $V_1$ and $V_3$, there is no need to direct outwards from $V_1$ and $V_3$, and thus the edges may be directed randomly.

(b) If there is no collider present in the path, but directing from either $V_1$ or $V_3$ outwards creates a new collider, then this collider will be preserved. The edge linked to the other node can then be directed randomly.

(c) If there is no collider present in the path, and directing from either node outwards does not create a collider, then both edges will be directed outwards from the independent nodes.

This procedure can easily be illustrated. See figure 3.1 for case (b) and figure 3.2 for case (c). In these figures $V_1$ is independent of $V_3$, and 3 edges have been added already.

In figure 3.1 the procedure of the algorithm is shown when a collider can be created by directing outwards from one node. Note that there is already an arc from $V_2$ to $V_4$, which must have been found due to the presence of a collider, which has been illustrated by adding node $V_5$. Firstly, the algorithm will direct outwards from node $V_1$. This does not result in a collider. Thereafter, it will direct outwards from $V_3$, which results in a collider. Therefore, this direction will be maintained. Note that either direction of the edge between $V_1$ and $V_2$ now results in the same (in)dependencies, and will therefore be chosen at random.

In figure 3.2 the algorithm tries to create colliders in the same way. This time however no collider can be created. The only possible way not to violate the independence is by directing outwards from both nodes. In the algorithm of Sneller, this was the standard procedure when a path between a pair of independent nodes had been found.
Observe that when there are multiple paths between $V_1$ and $V_3$, then these rules have to be applied *per path* and thus a collider must be present in every path. Otherwise the independence $V_1 \perp V_3$ would not be modeled.

It is important to note here that the paths that have been left undirected lie between dependent nodes. Instead of creating colliders, not creating colliders must now be the goal, because otherwise false independencies would be imposed. The last two rules are based on the IC and PC algorithm:

- If there is a structure $X \rightarrow Y \leftarrow Z$ ($X \not\perp Z$) and no arrowhead is pointing to $Z$, direct it as $X \rightarrow Y \rightarrow Z$.

- If there is a directed path $X \rightarrow Y \rightarrow Z$ and an edge between $X$ and $Z$, direct it as $X \rightarrow Z$.

Clearly, these rules do not contradict each other and do not create colliders or cycles. As a conclusion of this subsection, these steps are illustrated as well.

In figure 3.3 the left part shows rule 1, the right part shows rule 2. The upper row is the start situation and the lower row is the result of applying the respective rule.

Should there still be undirected edges, then we are left with two options: do not direct them, or direct them randomly. As the goal in this stage of the algorithm is to create no more colliders, and we do not want any undirected edges in the final BN, it is chosen to direct outwards from the node that has the most undirected edges left. This causes all other adjacent nodes to be become dependent. The other edges are then directed randomly.
3.3.2 Implementation

The entire algorithm is implemented in MATLAB. The main file has been programmed manually, but some functions we use have been made available by external sources. For example the Bayes Net Toolbox and the KCI-test of Zhang were already made available online. This gave the opportunity to use advanced mathematical codes that can only be written by very experienced MATLAB users. For the validation of the statistical assumptions, Uninet was used. This program, that has been developed by TU Delft, can be downloaded and may be used for free for academic purposes. In the beginning this program caused some problems as it is not installed on computers of the TU Delft and neither does it run on a virtual Windows machine, so users of Mac can have a hard time getting this program running.
Chapter 4

Results

In this chapter the results of the algorithm are described. It starts with a small section on the datasets, followed by a section on the validation of the normal copula. Hereafter the results are presented per dataset, based on a comparison with other algorithms. Each graphical network is analyzed with respect to the arcs present and their directions. Besides, an explanation of the variables and the correlation matrices of the datasets are given.

4.1 Datasets

As one of the goals is to compare the algorithm with other algorithms, the datasets were chosen on basis of availability of networks that a comparison could be made with. Margaritis (2005) has written a paper in which he introduces a method for learning the BN structure from data. Besides using his own algorithm, a comparison with the PC algorithm is present in the paper. The datasets considered are the Abalone and Housing dataset\(^4\). In the appendix an extended overview of the datasets is provided.

4.2 Validation of the normal copula

As has been mentioned, it is necessary to validate the assumption of the normal copula. This can be done easily by Uninet\(^5\). Uninet is a software program developed by the Risk and Decision Analysis group at the Delft Institute of Applied Mathematics. By creating a sample file from the data provided, Uninet offers the possibility to validate the normal copula and whether enough dependence has been modeled at the very end of the algorithm.

Unfortunately, Uninet rejects the assumption of the normal copula for both datasets. Though this might not sound very promising, it does not immediately imply that a non-parametric BN may not be used to model the data. The determinant test is known to be very strict and can reject the assumption too easily [10]. An explanation on these causes can be found in [13].

Another method to compare the DNR and DER is by checking the order of magnitude of the entrywise differences of the matrices. When these differences are of order \(10^{-2}\) at most, then the matrices are ‘sufficiently close’. There is actually no theorem to support this statement; rather it can be seen as a working solution. For the abalone dataset the maximum difference per matrix entry is 0.035, which gives us a reason to proceed. In the housing dataset the maximum difference is 0.125, but there are only 2 entries that exceed 0.10, namely 0.125

\(^4\)Both available from https://archive.ics.uci.edu/ml/datasets.html

\(^5\)Available from: http://www.lighttwist.net/wp/uninet
between industry and tax, and 0.110 between industry and rad. As ‘rad’ is an index value and thereby discrete, it is not taken into account in our actual model. The average difference of all entries is 0.05, which is still in an acceptable range. One might argue that the rejection of the normal copula by Uninet together with the exceedance of the entrywise differences of $10^{-2}$ provides us with enough reasons not to proceed with a non-parametric BN for the housing dataset, but as a good comparison can be made with the BNs from both [14] and [6] it is decided to keep the housing dataset.

4.3 Output of algorithm

The algorithm works as expected and runs without problems. To start with, an explanation of the variables and a graphical representation of the BNs produced is displayed. Thereafter an analysis of the BNs per dataset is given.

It is important to note that not all relations depicted by arcs need to be causal relations. Assume variable A and B are adjacent nodes in a BN. There could be a hidden third variable C causing both of them, which is called a latent variable. Although A and B are correlated, this does not imply causation. Another possibility is an interaction (e.g. biological) between these nodes.

4.3.1 Abalone dataset

The abalone dataset considers the age of abalones (sea shells) based on the number of their rings. There are 8 continuous variables and one discrete variable regarding the sex of the abalone. This variable will not be considered, as rank correlations with other variables will not mean anything, i.e. sex is no ordinal variable. There are 4177 joint samples and no missing values.

The names of the variables are mentioned in each node. A slightly more detailed explanation can be found in appendix A.

Clearly the BN can be split in two subparts: the six nodes that have adjacencies and the two nodes that are unconnected. It is interesting to note that height and rings are unconnected, whereas diameter has five adjacent nodes. The goal of the original researchers, who have produced this dataset, was to predict the age of abalones by their number of rings, as the relationship ‘age = number of rings + 1.5’ holds for the abalones investigated. Counting the
number of rings turned out to be a boring and time consuming task, which was the reason to search for the relationship between the number of rings and other properties of these animals. Therefore it is a disappointing conclusion that no direct relationship between rings - which is the number of rings - and the other variables has been found. However, this does not mean that there can be no (biological) relationship at all: there simply was no need to add these edges to represent enough dependence as captured by the determinant. Thus, the rank correlation between rings and the other variables does not exceed the rank correlation between the variables whose arcs have been added.

<table>
<thead>
<tr>
<th></th>
<th>Length</th>
<th>Diameter</th>
<th>Height</th>
<th>Whole weight</th>
<th>Shucked weight</th>
<th>Viscera weight</th>
<th>Shell weight</th>
<th>Rings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td>1.0000</td>
<td>0.9833</td>
<td>0.8882</td>
<td>0.9726</td>
<td>0.9568</td>
<td>0.9527</td>
<td>0.9479</td>
<td>0.6044</td>
</tr>
<tr>
<td>Diameter</td>
<td>0.9833</td>
<td>1.0000</td>
<td>0.8957</td>
<td>0.9713</td>
<td>0.9505</td>
<td>0.9484</td>
<td>0.9541</td>
<td>0.6229</td>
</tr>
<tr>
<td>Height</td>
<td>0.8882</td>
<td>0.8957</td>
<td>1.0000</td>
<td>0.9160</td>
<td>0.8742</td>
<td>0.9006</td>
<td>0.9212</td>
<td>0.6577</td>
</tr>
<tr>
<td>Whole weight</td>
<td>0.9726</td>
<td>0.9713</td>
<td>0.9160</td>
<td>1.0000</td>
<td>0.9771</td>
<td>0.9753</td>
<td>0.9694</td>
<td>0.6308</td>
</tr>
<tr>
<td>Shucked weight</td>
<td>0.9568</td>
<td>0.9505</td>
<td>0.8742</td>
<td>0.9771</td>
<td>1.0000</td>
<td>0.9476</td>
<td>0.9177</td>
<td>0.5394</td>
</tr>
<tr>
<td>Viscera weight</td>
<td>0.9527</td>
<td>0.9484</td>
<td>0.9006</td>
<td>0.9753</td>
<td>0.9476</td>
<td>1.0000</td>
<td>0.9381</td>
<td>0.6143</td>
</tr>
<tr>
<td>Shell weight</td>
<td>0.9479</td>
<td>0.9541</td>
<td>0.9212</td>
<td>0.9694</td>
<td>0.9177</td>
<td>0.9381</td>
<td>1.0000</td>
<td>0.6925</td>
</tr>
<tr>
<td>Rings</td>
<td>0.6044</td>
<td>0.6229</td>
<td>0.6577</td>
<td>0.6308</td>
<td>0.5394</td>
<td>0.6143</td>
<td>0.6925</td>
<td>1.0000</td>
</tr>
</tbody>
</table>

Table 4.1: Empirical rank correlation matrix for the abalone dataset

This can also be seen in table 4.1. The rank correlations of ‘rings’ roughly lie between 0.5 and 0.7, whereas the other variables have rank correlations greater than 0.8.

However, as the research goal is generally known in advance, there is way to overcome this by constructing what we would like to call a prepared graph. This means that one would add certain arcs in advance of running the algorithm. In this case, one might consider adding arcs between rings and for example length and whole weight. This will result in a more complex model, as the edges connected to rings have a lower rank correlation than the ones the algorithm has added, and therefore they cannot simply be interchanged while preserving enough dependence in terms of the determinant. In this situation, the determinant might be considered not to be a good stopping test because it does not result in a BN suitable for achieving the original research goals. Another consideration might be not to take all variables into account and for example remove (a subset of) shell weight, shucked weight and viscera weight. This simplifies the model while still taking the most important variables into account.

Referring to Sneller’s thesis, one of the problems she encountered was that no colliders were found in the dataset she had used. Unfortunately, no colliders were found in this dataset either. Although some conditional dependencies were found, no unconditional independencies were found. This has an important consequence, as this means that a big part of the algorithm is skipped, namely the procedure for directing edges that are adjacent to independent nodes. Therefore, the entire BN heavily relies on the randomized direction of the arcs. In fact, no unique BN can be produced, as multiple alternatives represent the same (in)dependencies, i.e. they belong to the same Markov equivalence class. On the one hand this is inherent to working
with BNs, but on the other hand one would prefer to have at least certainty about the direction of some edges.

What does look good is that length seems to be influenced by no other variables, but rather it has an influence on different kind of weights and the diameter of the sea shells. Although there is no proof that this is correct, it corresponds with the biological intuition that length influences weight.

Secondly, it can be seen that the whole weight is influenced by three other kinds of weight, the length and the diameter. Again this nicely corresponds with one’s biological intuition regarding these variables. Therefore it seems that the direction of the arcs is fairly correct, although no conclusions can be drawn with respect to the variables viscera weight and shell weight. For a more detailed explanation on these nodes, please check the appendix.

Although it is of great importance to model independence in the right way, it is also interesting to check whether the weaker relationships of dependencies are violated. Note that on the nodes viscera weight, diameter and whole weight multiple collider-like structures can be found. By this we mean we a collider structure whose top nodes are connected or they have a common parent. If this is not the case, i.e. there is a real collider, then the data dependencies would be violated. For example, check the diameter node. Three collider-like structures can be seen: (shucked weight, length), (shell weight, length) and (shell weight, shucked weight).

Shucked weight and length have an arc between them, shell weight and length have an arc between them as well, and shell weight and shucked weight have the common parent (length). This means that these collider-like structures are allowed to be present and they do not violate any dependencies. In fact, all collider-like structures present in the BN do not model two nodes as being mutually independent, which is a positive result.

Finally, a comparison with [6] will be made now. The output of the new algorithm is placed next to Margiritis’s BN for convenience.

![Figure 4.2: Output of the new algorithm (same as figure 4.1)](image1)

![Figure 4.3: Output of PC algorithm according to Margaritis [6]](image2)

There are two interesting differences: the number of edges and the adjacencies of height and rings. Whereas they were unconnected in our algorithm, now height has a connection to shell weight and rings is connected to shell weight and shucked weight. This could be explained by the way the PC algorithm constructs a BN: instead of adding edges based on a high rank correlation, it removes edges from a saturated graph based on conditional independencies. This
process terminates when all possible pairs of random variables have been tested for conditional independence given all other variables. There is however no guarantee that the result will be a BN: edges might be undirected (which is not the case in this BN) or they might be bidirected (as are shell weight - rings and shucked weight - rings). This happens when the assumptions of the PC algorithm fail [6]. The graph produced by Margaritis is actually no BN!

Another point of interest is the number of edges in both BNs. 14 edges had been added to our BN to represent enough dependence, whereas 9 were added using the PC algorithm. This difference could again be explained by the stopping test for adding edges: where our algorithm gained multiple successes in the stopping test after adding 14 edges - see figure 4.4 - the PC algorithm stops removing edges when all conditional independencies have been checked. Apparently, this results in less edges and therefore a simpler BN is found. The maximum number of arcs that can be added in a dag is \( \frac{n(n-1)}{2} \), which can easily be proven. Assume there are \( N \) nodes. As the first node \( N_1 \) can have \( N - 1 \) arcs, any second node \( N_2 \) can have \( N - 2 \) arcs (not pointing to itself nor to \( N_1 \)). Continuing this way to the last node, \( N_n \) can have 0 arcs added. Thus, in total there can be \( N - 1 + N - 2 + \ldots + 1 + 0 = \frac{N(N-1)}{2} \) arcs. Having 8 nodes, this number equals 28, which means our algorithm has added 50% of the number of arcs possible. The PC algorithm has added 32% of the total amount possible.

As can be seen in the figure 4.4, the DNR lies between the 0.05 and 0.1 percentile of the DBBN distribution. Based on the determinant test, a better fit could be made by adding even more edges to our graph. This does not necessarily imply that adding more edges will create a better BN: as the number of parents per node grows the conditioning set of the conditional density functions increases, as does the number of data required to quantify the graph. The BN that was produced by the PC algorithm does not pass the determinant stopping test, as can be seen in figure 4.5.

Finally, we want to note that the two graphs are not in the same Markov equivalence class. In the BN produced by the new algorithm, there is an arc between shell weight and viscera weight and a path shell weight \( \rightarrow \) whole weight \( \rightarrow \) viscera weight. In Margaritis’ graph, only
the latter path is present, and thus shell weight is modeled to be conditionally independent of viscera weight, given whole weight. The KCI-test rejects this conditional independence.

### 4.3.2 Housing dataset

The housing dataset considers the value of houses in downtown Boston. According to the dataset description, 13 continuous variables and 1 binary (dummy) variable form this dataset. However, 3 of the 13 continuous variables can be considered to be discrete or ‘semi-discrete’, and therefore they will not be taken into account. In the BNs constructed in [6] and [14] these nodes do not have any adjacencies, which supports the decision to disregard these variables. For every variable 506 joint samples are available and there are no missing values.

An advantage of this dataset is that it concerns familiar variables and therefore one can easily verify (or at least form an opinion) whether certain arcs in the BN of this dataset correspond to one’s intuition⁶.

![Figure 4.6: BN of the housing dataset resulting from the algorithm](image)

<table>
<thead>
<tr>
<th>CRIME</th>
<th>Crime rate per town</th>
</tr>
</thead>
<tbody>
<tr>
<td>INDUSTRY</td>
<td>Proportion of non-retail business acres per town</td>
</tr>
<tr>
<td>NOX</td>
<td>Nitric oxides concentration</td>
</tr>
<tr>
<td>ROOMS</td>
<td>Average number of rooms per dwelling</td>
</tr>
<tr>
<td>AGE</td>
<td>Proportion of owner-occupied units built prior to 1940</td>
</tr>
<tr>
<td>DISTANCE</td>
<td>Weighted distances to five Boston employment centres</td>
</tr>
<tr>
<td>TAX</td>
<td>Full-value property-tax rate per $10,000</td>
</tr>
<tr>
<td>B</td>
<td>$1000(Bk - 0.63)^2$ where Bk is the proportion of blacks by town</td>
</tr>
<tr>
<td>LOWER STATUS</td>
<td>% Lower status of the population</td>
</tr>
<tr>
<td>VALUE</td>
<td>Median value of owner-occupied homes in $1000's</td>
</tr>
<tr>
<td>ZN</td>
<td>Proportion of residential land zoned for lots over 25,000 sq.ft.</td>
</tr>
<tr>
<td>RIVER</td>
<td>Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)</td>
</tr>
<tr>
<td>TEACHER</td>
<td>Pupil-teacher ratio by town</td>
</tr>
<tr>
<td>RAD</td>
<td>Index of accessibility to radial highways</td>
</tr>
</tbody>
</table>

Table 4.2: Variables of the housing dataset. The lower four rows are not taken into account here, but they appear in the other papers and figures.

⁶Note that we explicitly do not say that we can verify whether arcs are correct.
The following table equals the empirical rank correlation matrix of the housing dataset. The names of the variables are only given per column in order the fit the page. Abbreviations have been used not to increase the width of the columns: CR = crime, IN = industry, RV = river, NX = NOX, RM = rooms, AG = age, DS = distance, RD = RAD, TX = TAX, TP = teacher, STS = lower status and MV = value.

<table>
<thead>
<tr>
<th></th>
<th>CR</th>
<th>ZN</th>
<th>IN</th>
<th>RV</th>
<th>NX</th>
<th>RM</th>
<th>AG</th>
<th>DS</th>
<th>TX</th>
<th>TP</th>
<th>B</th>
<th>STS</th>
<th>MV</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR</td>
<td>1.00</td>
<td>0.57</td>
<td>0.74</td>
<td>0.04</td>
<td>0.82</td>
<td>0.31</td>
<td>0.70</td>
<td>0.74</td>
<td>0.73</td>
<td>0.73</td>
<td>0.73</td>
<td>0.47</td>
<td>0.36</td>
</tr>
<tr>
<td>ZN</td>
<td>0.57</td>
<td>1.00</td>
<td>0.64</td>
<td>0.04</td>
<td>0.63</td>
<td>0.36</td>
<td>0.54</td>
<td>0.61</td>
<td>0.28</td>
<td>0.37</td>
<td>0.45</td>
<td>0.16</td>
<td>0.49</td>
</tr>
<tr>
<td>IN</td>
<td>0.74</td>
<td>0.64</td>
<td>1.00</td>
<td>0.09</td>
<td>0.79</td>
<td>0.42</td>
<td>0.68</td>
<td>0.76</td>
<td>0.46</td>
<td>0.66</td>
<td>0.43</td>
<td>0.29</td>
<td>0.64</td>
</tr>
<tr>
<td>RV</td>
<td>0.04</td>
<td>0.04</td>
<td>0.09</td>
<td>1.00</td>
<td>0.07</td>
<td>0.06</td>
<td>0.07</td>
<td>0.08</td>
<td>0.02</td>
<td>0.04</td>
<td>0.14</td>
<td>0.04</td>
<td>0.05</td>
</tr>
<tr>
<td>NX</td>
<td>0.82</td>
<td>0.63</td>
<td>0.79</td>
<td>0.07</td>
<td>1.00</td>
<td>0.31</td>
<td>0.80</td>
<td>0.88</td>
<td>0.59</td>
<td>0.65</td>
<td>0.39</td>
<td>0.30</td>
<td>0.64</td>
</tr>
<tr>
<td>RM</td>
<td>0.31</td>
<td>0.36</td>
<td>0.42</td>
<td>0.06</td>
<td>0.31</td>
<td>1.00</td>
<td>0.28</td>
<td>0.26</td>
<td>0.11</td>
<td>0.27</td>
<td>0.31</td>
<td>0.05</td>
<td>0.64</td>
</tr>
<tr>
<td>AG</td>
<td>0.70</td>
<td>0.54</td>
<td>0.68</td>
<td>0.07</td>
<td>0.80</td>
<td>0.28</td>
<td>1.00</td>
<td>0.80</td>
<td>0.42</td>
<td>0.53</td>
<td>0.36</td>
<td>0.23</td>
<td>0.66</td>
</tr>
<tr>
<td>DS</td>
<td>0.74</td>
<td>0.61</td>
<td>0.76</td>
<td>0.08</td>
<td>0.88</td>
<td>0.26</td>
<td>0.80</td>
<td>1.00</td>
<td>0.50</td>
<td>0.57</td>
<td>0.32</td>
<td>0.25</td>
<td>0.56</td>
</tr>
<tr>
<td>TX</td>
<td>0.73</td>
<td>0.28</td>
<td>0.46</td>
<td>0.02</td>
<td>0.59</td>
<td>0.11</td>
<td>0.42</td>
<td>0.50</td>
<td>1.00</td>
<td>0.70</td>
<td>0.32</td>
<td>0.28</td>
<td>0.39</td>
</tr>
<tr>
<td>TP</td>
<td>0.73</td>
<td>0.37</td>
<td>0.66</td>
<td>0.04</td>
<td>0.65</td>
<td>0.27</td>
<td>0.53</td>
<td>0.57</td>
<td>0.70</td>
<td>1.00</td>
<td>0.45</td>
<td>0.33</td>
<td>0.53</td>
</tr>
<tr>
<td>B</td>
<td>0.47</td>
<td>0.45</td>
<td>0.43</td>
<td>0.14</td>
<td>0.39</td>
<td>0.31</td>
<td>0.36</td>
<td>0.32</td>
<td>0.32</td>
<td>0.45</td>
<td>1.00</td>
<td>0.07</td>
<td>0.47</td>
</tr>
<tr>
<td>STS</td>
<td>0.36</td>
<td>0.16</td>
<td>0.29</td>
<td>0.04</td>
<td>0.30</td>
<td>0.05</td>
<td>0.23</td>
<td>0.25</td>
<td>0.28</td>
<td>0.33</td>
<td>0.07</td>
<td>1.00</td>
<td>0.21</td>
</tr>
<tr>
<td>MV</td>
<td>0.63</td>
<td>0.49</td>
<td>0.64</td>
<td>0.05</td>
<td>0.64</td>
<td>0.64</td>
<td>0.66</td>
<td>0.56</td>
<td>0.39</td>
<td>0.53</td>
<td>0.47</td>
<td>0.21</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>0.56</td>
<td>0.44</td>
<td>0.58</td>
<td>0.14</td>
<td>0.56</td>
<td>0.63</td>
<td>0.55</td>
<td>0.45</td>
<td>0.35</td>
<td>0.56</td>
<td>0.56</td>
<td>0.19</td>
<td>0.85</td>
</tr>
</tbody>
</table>

Table 4.3: Empirical rank correlation matrix for the housing dataset

Note that for the housing dataset again two variables remain unconnected: B and rooms. B is a factor related to the proportion of black people living per town and seems to be unrelated to the other variables. The number of rooms per dwelling is most times directly related to the housing price, and therefore to ‘value’.

An important observation is that age and value form a collider with ‘lower status’. However, this collider structure cannot be found in the dataset, and as reversing this arc does not create a cycle, it is chosen to have the direction lower status → value instead of lower status ← value. This results in the BN depicted in figure 4.7. Note that reversing this direction implies the relationship age ⊥ value | lower status. The independence test gives a p-value of order $10^{-8}$, which implies that this relationship is not supported by the data. However, we feel that
incorrectly modeling conditional independence is better than incorrectly modeling unconditional independence.

For a comparison with [6], his resulting BN is displayed as well in figure 4.8. This BN is produced by the PC algorithm in combination with a new independence test, called the recursive median algorithm (RM). In this figure the variables ‘ZN’, ‘river’, ‘teacher’ and ‘RAD’ appear, but only RAD has adjacent nodes.

Note that 4 nodes are left unconnected. B has no adjacencies in our algorithm either, and ZN, river and teacher are not taken into account at all. The decision to disregard these variables is now supported by figure 4.8. The correlation matrix only supports the removal of ‘river’, which does not have correlations higher than 0.14.

Also note that the industry variable has 5 parental nodes in our BN, whereas it has only one parent in the other BN. It can be seen that crime has 5 children in our network, and 4 adjacent nodes in that of PC algorithm. Of these four, the arcs to NOX and industry are present in our network as well, whereas RAD is one of the variables that was not taken into account and the edge to value was not found by our algorithm. Although both algorithms agree on the presence of an arc from crime to NOX, this one should be doubted because there seems to be no link between the rate of crime and the concentration of NOx in the air. Also note there is no arc between NOX and VALUE in either BN. This corresponds with the findings of the original researchers who concluded that there is no direct link between the value of houses and the concentration NOx.

In the graph of the PC algorithm a few edges remain bidirected or undirected, which means that again no actual BN is produced. There is an undirected edge between NOX and distance and a among others a bidirected arc between age and lower status. The former is in our BN directed from distance to NOX, which seems to be logical as a greater distance to employment centres leads to more demand of transportation and thereby to more NOx; the latter is in our network directed from age to lower status. Whether this is the right direction is hard to decide as no simple explanation can be found. Therefore no conclusion regarding the right direction of this arc can be drawn and it seems that a random choice works well.

The two other bidirected arcs are those between crime and RAD and between crime and value. RAD is not taken into account by us, as this variable is discrete, which could be the cause of the failure of the PC algorithm in combination with the RM-test to determine the direction of this arc. The other one does not show up in our BN, but there is a path crime → age → lower status → value, which suggests to direct this arc as crime → value.

With respect to the collider-like structures, it can again be concluded that our algorithm does not create any invalid independencies (after reversing the arc between lower status and value). This conclusion can not be drawn as easily in the case of the PC algorithm due to the bidirected edges. There is a way of directing the bidirected edges such that no colliders are formed, but when for example crime ↔ RAD is directed as crime → RAD, then crime and tax form a collider with RAD. The same holds for the triple rooms, value and crime. This does not correspond with the data, because no independencies were found.

As was the case with the abalone dataset, the number of edges added by our algorithm exceeds the number of the PC algorithm: 15 were added by us, where PC added 10. Interestingly, this is approximately the same ratio as before, which was 14:9. In the saturated graph, 45 arcs would have been present (based on 10 nodes).
Finally, two more BNs that have been constructed with the housing dataset are available, with which a comparison will be made. These graphs are presented in the figure 4.9 and figure 4.10, based on a figure in [14].

Figure 4.9: BN of the housing dataset from the PC algorithm with the KCI test

Figure 4.10: BN of the housing dataset from the PC algorithm with the CI$_{perm}$ test

In the article, Zhang compares his KCI-test with a different conditional independence test. As the KCI-test was also used in the algorithm of this thesis, it is interesting to make a comparison with this one. However, it must also be noted that there are some differences as well between the BN displayed in figure 4.8 and the one based on the KCI-test in figure 4.9. For example, the doubtful arc between crime and NOX does not appear in figure 4.9, but instead there now is an edge between tax and B, which does not seem quite logical either. This edge is undirected, which might be an indication of the incorrectness of adding this edge. As we suggested earlier, the bidirected edge between crime and value has now been directed as crime $\rightarrow$ value.

In figure 4.9, rooms is connected to value and lower status, which seem to be correct relations. B shares an undirected edge with tax, but as mentioned before this one can be doubted. RAD, teacher, river and ZN do not have any adjacencies (in both algorithms of Zhang), which supports our decision not to take these variables into account (and we have not displayed them here). The arc lower status $\rightarrow$ value is present here, which supports our decision to reverse this arc directly after the execution of the algorithm. Lower status and age now point towards value as well, while the former was not present in figure 4.8 and the latter was directed the other way around.

Lastly, it has to be noted that changing the independence test while sticking to the same algorithm greatly affects the output of the algorithm. The graphs of figure 4.9 and 4.10 only share 2 arcs, while there are 11 edges/arcs in figure 4.9 and 5 in figure 4.10. It is therefore not surprising that these graphs are no elements of the same Markov equivalence class, as (for example) NOX and age are dependent when using the KCI test, but they become independent when using the CI$_{perm}$ test. In fact, the four graphs belonging to the housing dataset that were shown in this section are all elements of a different Markov equivalence class. The determinant test indicated that only our BN represents enough dependence, and rejected the other graphs. The direction of the undirected and bidirected edges did not influence this result. The figure on the following page provides a global overview of all outputs.
(a) Output of the new algorithm (KCI-test)

(b) Output of PC algorithm (RM-test)

(c) Output of PC algorithm (KCI-test)

(d) Output of PC algorithm (CI\textsubscript{perm}-test)

Figure 4.11: Overview of the graphs that represent the housing dataset. Figure (a) and (c) share the same independence test, but have a different algorithm. Figure (b), (c) and (d) share the same algorithm, but have a different independence test.
4.4 Validation of algorithm

As no independencies were found in the previous datasets, no conclusion has been drawn about the performance of the algorithm when independent nodes have to be taken into account. Therefore, we will now consider an artificial dataset in which independencies are present. Uninet does not reject the normal copula, and thus a non-parametric BN can be used as a suitable model. Using the KCI test, the following independencies are found:

\[ 1 \perp \{2, 3, 6, 7\}, 2 \perp \{1, 3, 6, 7\}, 3 \perp \{2, 6\}, 4 \perp 6 \text{ and } 5 \perp 6. \]

In table 4.4 it can be seen that these independencies correspond to pairs of variables that have low rank correlations (\(\leq 0.04\)). The result of the algorithm - after a successful stopping test - is given in the figure below.

![Figure 4.12: BN of the artificial dataset](image)

The algorithm seems to work well with respect to the independent nodes. One can easily verify that with exception of \(2 \perp 7\) all independencies are modeled in the BN. Given the edges present in this BN, it not possible to direct them such that no independence is violated. The only way to model \(V_2\) independent of \(V_7\) while maintaining these arcs is by changing \(V_5 \rightarrow V_7\) to \(V_5 \leftarrow V_7\). This however violates \(V_5 \perp V_6\). Therefore it can be concluded that the algorithm maintains independencies as long as possible.

The arcs between \(V_4\) and \(V_8\), \(V_4\) and \(V_5\), \(V_5\) and \(V_8\), and \(V_7\) and \(V_8\) cannot be learnt from the independencies. Indeed they differ per run of the algorithm, and thus randomness has been implemented correctly.

<table>
<thead>
<tr>
<th>(V_1)</th>
<th>(V_2)</th>
<th>(V_3)</th>
<th>(V_4)</th>
<th>(V_5)</th>
<th>(V_6)</th>
<th>(V_7)</th>
<th>(V_8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(V_1)</td>
<td>1</td>
<td>0.0301</td>
<td>0.0253</td>
<td>0.4937</td>
<td>0.3637</td>
<td>0.0104</td>
<td>0.0141</td>
</tr>
<tr>
<td>(V_2)</td>
<td>0.0301</td>
<td>1</td>
<td>0.0211</td>
<td>0.7474</td>
<td>0.5727</td>
<td>0.0060</td>
<td>0.0097</td>
</tr>
<tr>
<td>(V_3)</td>
<td>0.0253</td>
<td>0.0211</td>
<td>1</td>
<td>0.3472</td>
<td>0.3956</td>
<td>0.0191</td>
<td>0.6740</td>
</tr>
<tr>
<td>(V_4)</td>
<td>0.4937</td>
<td>0.7474</td>
<td>0.3472</td>
<td>1</td>
<td>0.7954</td>
<td>0.0063</td>
<td>0.2358</td>
</tr>
<tr>
<td>(V_5)</td>
<td>0.3637</td>
<td>0.5727</td>
<td>0.3956</td>
<td>0.7954</td>
<td>1</td>
<td>0.0001</td>
<td>0.2672</td>
</tr>
<tr>
<td>(V_6)</td>
<td>0.0104</td>
<td>0.0060</td>
<td>0.0191</td>
<td>0.0063</td>
<td>0.0001</td>
<td>1</td>
<td>0.3175</td>
</tr>
<tr>
<td>(V_7)</td>
<td>0.0141</td>
<td>0.0097</td>
<td>0.6740</td>
<td>0.2358</td>
<td>0.2672</td>
<td>0.3175</td>
<td>1</td>
</tr>
<tr>
<td>(V_8)</td>
<td>0.3053</td>
<td>0.4729</td>
<td>0.5337</td>
<td>0.7270</td>
<td>0.9015</td>
<td>0.0959</td>
<td>0.5226</td>
</tr>
</tbody>
</table>

Table 4.4: Correlation matrix of the artificial dataset
The data that was used to learn the BN of figure 4.12 from, was sampled from the BN in figure 4.14. For an easy comparison, our output has also been placed on the same page. This dataset involves 8 continuous variables and 5000 joint samples, which means it is comparable to the abalone dataset (8 continuous variables and about 4200 joint samples).

Note that there are 9 arcs in the original model, and 16 in the BN produced by our algorithm. This is the same ratio as found in the abalone dataset, and supports that our algorithm tends to add a relatively high number of arcs. Of these 9 arcs, 8 have been directed equally, while $V_4 \leftarrow V_5$ was originally directed as $V_4 \rightarrow V_5$. Due to the many arcs pointing to $V_4$ and $V_5$ in our BN, this particular arc can be changed without affecting the (conditional) independencies modeled, i.e. the graph still belongs to the same Markov equivalence class. In the original BN this is not possible, as the relations $V_1 \perp V_5$ and $V_2 \perp V_5$ would be implied.
Chapter 5

Conclusion and discussion

The last chapter of this thesis has multiple purposes. The most important one is to draw conclusions from our research. Before doing so, a summary of the entire process that is described in this thesis is given. Furthermore a discussion on the results is included, as no research is performed perfectly, and new insights may have changed aspects in an earlier stage. The question whether the objectives of the research have been reached will be answered as well. Finally, some recommendations for further research are provided.

5.1 Conclusions

First of all a summary of the entire process is given by going through all steps of the algorithm. Then the conclusions are drawn based on our comparisons with the BNs of Margaritis and Zhang that involve the abalone and housing dataset.

5.1.1 Summary

The research started with the intention to create an algorithm for learning non-parametric BNs from data. The advantage of these BNs is that the only assumption is that the data follows the normal copula. They are able to deal with hybrid data, which consists of both discrete and continuous data, but perform best when only continuous variables are present.

The first step of the algorithm is to compute the empirical rank correlation matrix (ERCM) and an independence matrix. The ERCM contains the correlations between the ranks. A high rank correlation is the basis for adding edges to the network. However, they may never be added between independent nodes, because that would violate the independence. Therefore, it is always checked that no edges are added between independent nodes.

When enough edges have been added, they are directed according to rules that are based on the principle of d-separation. The first way to do this is by using colliders, i.e. two nodes that are independent, but become dependent given a third node. There is only one way to model this dependence structure, and therefore it is a great aid in directing edges.

It is important that there is no path - a sequence of directed edges - between independent nodes. One option is to direct all edges that are adjacent to the independent nodes and on a path between independent nodes outwards, i.e. not pointing to the independent nodes. However, this reduces the amount of possible BNs. Therefore it is checked whether there is already a collider on these paths or whether a collider can be constructed by directing edges.
outwards from one node only. If this is possible, then we make use of this and the other edge is directed randomly.

The edges that remain undirected lie between dependent nodes and it is not possible to learn their direction from the dataset they originate from. To direct them, two rules that are based on the IC and PC algorithm are applied. They prevent the creation of new colliders, which would incorrectly model independence, and make sure no cyclic graph is generated. Should there still be any undirected edges, their direction is chosen at random, making sure that the graph remains acyclic.

5.1.2 Comparison with Margaritis and Zhang

At first it must be noted that is difficult to draw conclusions about the performance of this algorithm, as it was only applied to relatively small datasets and compared with a few other BNs. In order to create a good comparison and draw some sound conclusions with respect to the performance of the algorithm, more datasets and BNs would be required. Not only does this require a lot more time, but it is a difficult task to get hold of more suitable data, especially in the scope of writing a bachelor’s thesis.

Nevertheless, the most noticeable differences will be used to at least give an indication about the correctness and performance of the algorithm. This concerns the number and directions of edges.

The most noticeable difference compared to Margaritis and Zhang is the number of edges. The algorithm of this paper on average adds 1.5 times more edges than the other algorithms. This has both its advantages and disadvantages, and one might say that this a trade-off: on the one hand there must be enough edges to represent the dependencies found in the dataset, but on the other hand less edges result in an easier to understand BN and smaller conditioning sets in the conditional densities. On a personal note, we feel that the algorithms that were used as a comparison have a better way to determine which edges should be present (or at least the number of them).

With respect to the direction of the arcs, the algorithm seems to do a good job. Especially in the abalone dataset the direction of the arcs intuitively seem to make sense. In the housing dataset there is a doubtful connection between the rate of crimes and the concentration of NOx in the air, but this one is also present in Margaritis’s BN of this dataset. In order to make a sound conclusion with respect to the direction of the arcs, more comparisons would have to be made and the directions would have to be judged by experts. Unfortunately, this falls outside the scope of this bachelor project.

Another difference is the fact that no edges remain undirected or bidirected after applying our algorithm. This is due to the last step of the algorithm that randomly decides the direction of the remaining undirected edges. Another way of doing this would be by asking experts for the direction of arcs. Obviously, this would require their availability, which might not always be the case. Therefore, a random choice seems to be a good solution in the absence of experts. A possible improvement might be to keep a list of the arcs whose direction was determined randomly, and possibly reverse doubtful directions. Especially when there are many arcs, this comes with the risk of creating a directed graph, which would lead to reversing even more arcs to prevent cycles. This unwanted effect could be prevented by directing some edges in advance of the algorithm. This guarantees that some directions are fixed, while one prevents the creation of cycles at the same time.
In order to assess the performance of the algorithm when independent variables are present in the dataset, an artificial dataset has been used. Of the 10 independencies found, only 1 was violated in the BN. This was the best result possible given the present arcs, because it turned out to be impossible not to violate at least one independence. The only way to solve this is by removing an arc, but then the stopping test would not be passed. This could be overcome by adding other arcs, but in advance it is unknown how many arcs must be added to ‘compensate’ for removing one arc.

5.2 Discussion

A discussion on the results is as important as the results itself. Although we have presented some results, there are aspects of the algorithm that no conclusion can be drawn about easily. The reason for this will be elaborated on in this section.

It must be noted that it is difficult to assess the direction of the arcs. On the one hand multiple BNs can represent the exact same dependencies (up to Markov equivalence classes). Reversing certain arcs can result in the same dependencies, but in a better model of causal relationships. This also means that there is no unique BN that the algorithm will produce for a certain dataset. Another run can (and will) result in a different direction for certain edges. Thus, drawing conclusions about the correct assignment of directions is no easy task. When comparing to other algorithms this is even a harder task, as only changing the independence test can result in a very different BN. Among other causes are the procedure of adding or removing edges and the stopping tests used.

It is important to keep in mind that there is no ‘best’ BN for a specific dataset. Therefore, having an expert to assess the BN will always be required and one should realize that relying on an algorithm only to produce a BN never is a good idea. To improve the usefulness of the BN, arcs may be added in advance, and certain variables may be disregarded. These choices must be made by the modeler and no general rule can be established here.

The last question to answer is whether the objective of this research has been reached. In the introduction it has been stated that the goal was to design a new algorithm for learning non-parametric BNs from data, while trying to improve on the problems Sneller had encountered. The design of a new algorithm has succeeded, and the procedure of direction edges between independent nodes has been improved. Based on the abalone and housing dataset, it was not possible to assess the algorithm when independent had to be taken into account. The artificial dataset supports that the algorithm correctly handles independence.

5.3 Recommendations

Having completed this research, some questions still need to be answered or might be examined in more detail. The following list provides a few suggestions:

- Investigate why the algorithm tends to add a relative high number of arcs, i.e. about 1.5 times more than the PC algorithm and the artificial dataset that was introduced. Check whether other stopping tests are suitable for this algorithm.
- In our research it was found that it might be useful not to start with nodes only, but create a prepared graph instead. This could for example be related to the goal of the research, or edges could be added based on expert judgment. Investigate the dataset and decide if all variables need to be taken into account.
Bibliography


Appendices
Appendix A

Details of datasets

This appendix provides a more detailed explanation of the datasets used.

Abalone

According to the dataset description, the goal of the original researchers was to predict the age of abalone from physical measurements. They write: “The age of abalone is determined by cutting the shell through the cone, staining it, and counting the number of rings through a microscope – a boring and time-consuming task. Other measurements, which are easier to obtain, are used to predict the age. Further information, such as weather patterns and location (hence food availability) may be required to solve the problem.”

The dataset consists of 9 variables and 4177 data points, with no missing data. One of the variables describes the sex of the abalone (male/female/infant) and was disregarded during our research as no ordering can be made.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Node name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td>LENG</td>
<td>Longest shell measurement [mm]</td>
</tr>
<tr>
<td>Diameter</td>
<td>DIAM</td>
<td>Perpendicular to length [mm]</td>
</tr>
<tr>
<td>Height</td>
<td>HEIG</td>
<td>With meat in shell [mm]</td>
</tr>
<tr>
<td>Whole weight</td>
<td>WHWE</td>
<td>Whole abalone [g]</td>
</tr>
<tr>
<td>Shucked weight</td>
<td>SHWE</td>
<td>Weight of meat [g]</td>
</tr>
<tr>
<td>Viscera weight</td>
<td>VIWE</td>
<td>Gut weight (after bleeding) [g]</td>
</tr>
<tr>
<td>Shell weight</td>
<td>SLWE</td>
<td>After being dried [g]</td>
</tr>
<tr>
<td>Number of rings</td>
<td>RING</td>
<td>+1.5 gives the age in years</td>
</tr>
</tbody>
</table>

Table A.1: Explanation of the abalone dataset

Housing

The housing dataset concerns housing values in the suburbs of Boston. There are 14 variables with 506 data points with no missing data. One of the variables is a binary dummy variable, and was not used for our research. Three other variables turned out to be discrete or ‘semi-discrete’ and were therefore not used either.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRI</td>
<td>Crime rate per town</td>
</tr>
<tr>
<td>IND</td>
<td>Proportion of non-retail business acres per town</td>
</tr>
<tr>
<td>NOX</td>
<td>Nitric oxides concentration</td>
</tr>
<tr>
<td>RM</td>
<td>Average number of rooms per dwelling</td>
</tr>
<tr>
<td>AGE</td>
<td>Proportion of owner-occupied units built prior to 1940</td>
</tr>
<tr>
<td>DIS</td>
<td>Weighted distances to five Boston employment centres</td>
</tr>
<tr>
<td>TAX</td>
<td>Full-value property-tax rate per $10,000</td>
</tr>
<tr>
<td>B</td>
<td>$1000(Bk - 0.63)^2$ where Bk is the proportion of blacks by town</td>
</tr>
<tr>
<td>LST</td>
<td>% Lower status of the population</td>
</tr>
<tr>
<td>MED</td>
<td>Median value of owner-occupied homes in $1000’s</td>
</tr>
<tr>
<td>ZN</td>
<td>Proportion of residential land zoned for lots over 25,000 sq.ft.</td>
</tr>
<tr>
<td>CHAS</td>
<td>Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)</td>
</tr>
<tr>
<td>PTRATIO</td>
<td>Pupil-teacher ratio by town</td>
</tr>
<tr>
<td>RAD</td>
<td>Index of accessibility to radial highways</td>
</tr>
</tbody>
</table>

Table A.2: Explanation of the housing dataset
Appendix B

MATLAB code

The main MATLAB code that was written by the author is displayed in this appendix. Linenumbers help to indicate whether a code line had to be broken to fit the page. Besides this code several other functions have been used, whose source has been mentioned in the thesis, but for an overview will be presented here as well. The PathFinder function has been edited slightly, as one of its input arguments was a matrix that contained all edges. This matrix was then used to construct the adjacency matrix, but in our main code the adjacency matrix was already built.

The extra packages include:

- Bayes Net Toolbox for MATLAB: https://code.google.com/p/bnt/

```matlab
clc
clear all

% Some functions from external sources are being used, for example the BNT
% package for MATLAB and the KCI-test. Their paths have been added to MATLAB manually.

% data = importdata('abalone.txt');
data = importdata('housing.txt');

% VALIDATION
n = size(data,2);  % number of variables (2 = #columns)
adj = zeros(n,n);  % empty adjacency matrix
dag = zeros(n,n);  % empty bbn
alpha = 0.001;     % p-value for KCI-test
numberarcs = 9;     % number of arcs that must be added
arcsadded = 0;      % number of added arcs
dir_added = 0;      % number of added directions

cr = corr(data, 'type', 'Spearman');  % Correlation matrix
```
% INITIAL PHASE

% Independence test
dep = zeros(n,n);
dep2 = zeros(n,n);
for i=1:n
    for j=i+1:n
        [p_val, stat] = indtest_new(data(:,i),data(:,j),[],[]);
        if p_val > alpha
            dep(i,j) = 1;
            dep(j,i) = 1;  % symmetrical
            dep2(i,j) = p_val;
            dep2(j,i) = p_val;
        end
    end
end

% Add arcs based on high correlation
cr2 = cr-eye(n);  % remove diagonal entries
while arcsadded < numberarcs
    [value, location] = max(cr2(:));  % find maximum value of cr2
    [R, C] = ind2sub(size(cr2), location);
    if dep(R,C) == 0  % Nodes are not independent
        adj(R,C) = 1;  % nodes are now adjacent
        adj(C,R) = 1;  % create symmetrical matrix
        arcsadded = arcsadded + 1;
    end
    cr2(R,C) = 0;  % remove entry from matrix
    cr2(C,R) = 0;  % remove symmetrical entry
end
disp('Initial phase finished, starting main phase.')

% MAIN PHASE

% Conditional independence test based on adjacent nodes
% First find conditional dependence
adj_count = sum(adj~=0,1);  % number of adjacencies per variable (column)
cond_dep = [];  % col. 1 cond. dep. of col. 2 given col. 3 per row
cond_count = 1;  % number of conditional dependencies+1
for i=1:n
    if adj_count(i) > 1  % at least two neighbours
        index = find(adj(:,i));  % index of neighbours (non−zero entries)
        for j=1:length(index)
            for k=j+1:length(index)
                [p_val, stat] = indtest_new(data(:,index(j)),data(:,index(k)),data(:,i),[]);
                if p_val < alpha  % conditionally dependent
cond_dep(cond_count,:) = [index(j) index(k) i];
end
cond_count = cond_count+1;
end
end
cond_count = cond_count-1;  % remove 1 that was required as index
disp('Conditional_independencies_found.')
% Find colliders
for i=1:cond_count
    if dep(cond_dep(i,1), cond_dep(i,2)) == 1  % collider found
        dag(cond_dep(i,1), cond_dep(i,3)) = 1;  % create collider
        dag(cond_dep(i,2), cond_dep(i,3)) = 1;
        dir_added = dir_added + 2;
        adj(cond_dep(i,1), cond_dep(i,3)) = 0;  % remove entry from adj
        adj(cond_dep(i,3), cond_dep(i,1)) = 0;
        adj(cond_dep(i,2), cond_dep(i,3)) = 0;
        adj(cond_dep(i,3), cond_dep(i,2)) = 0;
    end
end
% Find number of paths between independent nodes
% First create edges matrix – Not required since PathFinder2
index = find(adj);  % find edges of BBN
[R, C] = ind2sub(size(adj), index);
edges = zeros(arcsadded,2);  % matrix for path finding function
for i=1:arcsadded
    edges(i,:) = [R(i) C(i)];
end
% Now find number of paths between indep. nodes and apply rule
disp('Starting_finding_paths_between_independent_nodes.')
for i=1:n
    for j=i+1:n
        if dep(i,j) == 1  % nodes are independent
            if sum(find(adj(i,:)) > 0 && sum(find(adj(j,:))
                paths = PathFinder2(adj,i,j);  % All paths between node i and j
                number_paths = size(paths,1);  % #paths between node i and j
                for k=1:number_paths
                    collider_present = 0;  % collider present in path
                    if sum(paths(k,:)~=0) == 3  % number of non-zero elements (in path k) = 3
                        dag(paths(k,1), paths(k,2)) = 1;  % create collider
\[ \text{dag}(\text{paths}(k,3), \text{paths}(k,2)) = 1; \% \text{create}\]
\[ \text{collider}\]
\[ \text{dir}_\text{added} = \text{dir}_\text{added} + 2; \% \text{added two directions}\]
\[ \text{adj}(\text{paths}(k,1), \text{paths}(k,2)) = 0; \]
\[ \text{adj}(\text{paths}(k,2), \text{paths}(k,1)) = 0; \]
\[ \text{adj}(\text{paths}(k,3), \text{paths}(k,2)) = 0; \]
\[ \text{adj}(\text{paths}(k,2), \text{paths}(k,3)) = 0; \]
\[ \text{elseif} \sum(\text{paths}(k,:)) \neq 0 > 3 \% \text{more than three}\]
\[ \text{elements in path } k\]
\[ \text{length} = \sum(\text{paths}(k,:)) \neq 0; \% \text{length of path } k\]
\[ \text{for } l=2:\text{length}-1 \% \text{check middle nodes only for}\]
\[ \text{colliders - walk through path } k\]
\[ \text{if} \quad \text{dag}(\text{paths}(l-1), \text{paths}(1)) = 1 \&\& \text{dag}(\text{paths}\]
\[ \text{(1+1), paths}(1)) = 1 \% \text{collider found in}\]
\[ \text{path } k\]
\[ \text{coll\_pres} = \text{coll\_pres} + 1; \% \text{count}\]
\[ \text{colliders}\]
\[ \text{end}\]
\[ \text{end} \% \text{walked through pad}\]
\[ \text{if} \quad \text{coll\_pres} > 0 \% \text{collider present in path } \Rightarrow \]
\[ \text{direct outwards randomly}\]
\[ \text{dag}(\text{paths}(k,1), \text{paths}(k,2)) = \text{round}(\text{rand}); \% \]
\[ \text{random } 0 \text{ or } 1 \text{ for first edge}\]
\[ \text{dag}(\text{paths}(k,2), \text{paths}(k,1)) = \text{abs}(\text{dag}(\text{paths}\]
\[ \text{(k,1), paths}(k,2)) - 1); \% 1 \rightarrow 0 \text{ and } 0 \rightarrow 1\]
\[ \text{dag}(\text{paths}(k,\text{length}-1), \text{paths}(k,\text{length})) =\]
\[ \text{round}(\text{rand}); \% \text{random } 0 \text{ or } 1 \text{ for last edge}\]
\[ \text{dag}(\text{paths}(k,\text{length}), \text{paths}(k,\text{length}-1)) = \text{abs}\]
\[ (\text{dag}(\text{paths}(k,\text{length}-1), \text{paths}(k,\text{length})) - 1); \% 1 \rightarrow 0 \text{ and } 0 \rightarrow 1\]
\[ \text{dir\_added} = \text{dir\_added} + 2; \]
\[ \text{adj}(\text{paths}(k,1), \text{paths}(k,2)) = 0; \% \text{remove}\]
\[ \text{entries from adjacency matrix}\]
\[ \text{adj}(\text{paths}(k,2), \text{paths}(k,1)) = 0; \]
\[ \text{adj}(\text{paths}(k,\text{length}-1), \text{paths}(k,\text{length})) = 0; \]
\[ \text{adj}(\text{paths}(k,\text{length}), \text{paths}(k,\text{length}-1)) = 0; \]
\[ \text{else} \% \text{Path} > 3, \text{no collider}\]
\[ \text{if} \quad \text{dag}(\text{paths}(k,3), \text{paths}(k,2)) = 1 \% \]
\[ \text{collider found 'near' first node}\]
\[ \text{dag}(\text{paths}(k,1), \text{paths}(k,2)) = 1; \% \text{direct}\]
\[ \text{first edge outwards}\]
\[ \text{dag}(\text{paths}(k,\text{length}-1), \text{paths}(k, \text{length})) =\]
\[ \text{round}(\text{rand}); \% \text{direct other edge}\]
\[ \text{randomly}\]
\[ \text{dag}(\text{paths}(k,\text{length}), \text{paths}(k, \text{length}-1)) = \text{abs}(\text{dag}(\text{paths}(k,\text{length}-1), \text{paths}(k, \text{length})) - 1); \]
\[ \text{dir\_added} = \text{dir\_added} + 2; \]
\[ \text{adj}(\text{paths}(k,1), \text{paths}(k,2)) = 0; \% \text{remove}\]
entries from adjacency matrix

adj(paths(k,2), paths(k,1)) = 0;
adj(paths(k,length-1), paths(k,length)) = 0;
adj(paths(k,length), paths(k,length-1)) = 0;

elseif dag(paths(k,length-2), paths(k,length -1)) == 1
% collider found 'near' last node

dag(paths(k,length), paths(k,length-1)) = 1; % direct last edge outwards

dag(paths(k,1), paths(k,2)) = round(rand) ; % direct other edge randomly

dag(paths(k,2), paths(k,1)) = abs(dag(paths(k,1), paths(k,2))-1);
dir_added = dir_added + 2;

adj(paths(k,1), paths(k,2)) = 0; % remove entries from adjacency matrix

adj(paths(k,2), paths(k,1)) = 0;
adj(paths(k,length-1), paths(k,length)) = 0;
adj(paths(k,length), paths(k,length-1)) = 0;

else % directing outwards does not create collider

dag(paths(k,1), paths(k,2)) = 1;
dag(paths(k,length), paths(k, length-1)) = 1;
dir_added = dir_added + 2;

adj(paths(k,1), paths(k,2)) = 0; % remove entries from adjacency matrix

adj(paths(k,2), paths(k,1)) = 0;
adj(paths(k,length-1), paths(k,length)) = 0;
adj(paths(k,length), paths(k,length-1)) = 0;

end
end
end
end

disp('Finished finding paths between independent nodes.')

% Apply rules based on IC and PC

% Rule 1: A→B→C becomes A→B→C

for i=1:n
    index = find(dag(i,:)); % index of directed neighbours (A→B)
for j = 1:size(index, 1)
    if size(index, 2) > 0 % not the empty matrix
        index2 = find(adj(index(j,:),:)); % index of undirected neighbours of B (= index(j))
        for k = 1:size(index2, 1)
            if index2(k) == i == 1 % adjacent node not equal to A
                if sum(dag(:, index2(k))) == 0 % no D->C
                    dag(index(j), index2(k)) = 1;
                    dir_added = dir_added + 1;
                    adj(index(j), index2(k)) = 0; % remove entry from adjacency
                    adj(index2(k), index(j)) = 0;
                end
            end
        end
    end
end

disp('Applied rule 1 of IC/PC.')

% Rule 2: A->B->C and A->C gives A->C
for i = 1:n
    index = find(dag(i,:)); % index of directed neighbours (A->B)
    for j = 1:size(index, 1)
        if size(index, 2) > 0
            index2 = find(dag(index(j,:),:)); % index of directed neighbours of B (= index(j))
            for k = 1:length(index2)
                if adj(i, index2(k)) == 1 % A-C present
                    dag(i, index2(k)) = 1; % A->C
                    dir_added = dir_added + 1;
                    adj(i, index2(k)) = 0; % remove entry from adjacency
                    adj(index2(k), i) = 0;
                end
            end
        end
    end
end

disp('Applied rule 2 of IC/PC.')

% Direct outwards from the node with the most remaining undirected edges.
% then direct all other remaining nodes randomly.
final_undir = sum(adj == 0, 2); % Vector with number of remaining undirected edges per node.
[C, I] = max(final_undir); % Value and index of node with the maximum # undirected edges
for i = 1:n
if \( \text{adj}(I, i) == 1 \);
\[
\text{dag}(I, i) = 1;
\]
if \( \text{acyclic}(\text{dag}) == 1 \) % No cycle created
\[
\text{adj}(I, i) = 0;
\]
\[
\text{adj}(i, I) = 0;
\]
else % Cycle was created
\[
\text{dag}(I, i) = 0;
\]
\[
\text{dag}(i, I) = 1; % Reverse direction
\]
\[
\text{adj}(I, i) = 0;
\]
\[
\text{adj}(i, I) = 0;
\]
end
\[
\text{dir}_{\text{added}} = \text{dir}_{\text{added}} + 1;
\]
end

% Direct last remaining edges randomly
for \( i = 1:n \)
for \( j=i+1:n \)
if \( \text{adj}(i, j) == 1 \)
\[
\text{dag}(i, j) = \text{round} \left( \text{rand} \right);
\]
\[
\text{dag}(j, i) = \text{abs} (\text{dag}(i, j) - 1);
\]
if \( \text{acyclic}(\text{dag}) == 1 \) % No cycle created
\[
\text{adj}(i, j) = 0;
\]
\[
\text{adj}(j, i) = 0;
\]
else
\[
\text{dag}(i, j) = \text{dag}(j, i); % Reverse direction
\]
\[
\text{dag}(j, i) = \text{abs} (\text{dag}(i, j) - 1);
\]
\[
\text{adj}(i, j) = 0;
\]
\[
\text{adj}(j, i) = 0;
\]
end
\[
\text{dir}_{\text{added}} = \text{dir}_{\text{added}} + 1;
\]
end
end