CHALMERS
UNIVERSITY OF TECHNOLOGY

Bayesian Network Approach for Modelling and Inference of Communication Networks

Master's thesis in Engineering Mathematics and Computational Science

Themis Mouliakos

Department of Mathematics
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Master's thesis 2019

Bayesian Network Approach for Modelling and Inference of Communication Networks

Themistoklis Christos Mouliakos

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematics
Chalmers University of Technology
Gothenburg, Sweden 2019

Bayesian Network Approach for Modelling and Inference of Communication Networks
Themistoklis Christos Mouliakos

Student: Themistoklis Christos Mouliakos
Supervisor/Examiner: Larisa Beilina, Chalmers University of Technology

Master's Thesis 2019
Department of Mathematics
Chalmers University of Technology
SE-412 96 Gothenburg, Sweden
Telephone +46 (0)31 772 1000

Gothenburg, Sweden 2019

Abstract

A collection of radio base stations defines the radio access network which is responsible for connecting the user equipment with the core network which in turn connects to the internet through a gateway. The radio base station provides the uplink and downlink communication of the user equipment and its cover region is divided in cells. There can be thousands of parameters affecting the functionality of a radio base station and a primary concern of this thesis is to estimate probabilistic relations between those parameters and the key performance indicators.

Bayesian Networks is a powerful mathematical tool which can model complex systems and present possible co-influences between variables. In the last decades there was a great advancement on creating Bayesian Networks from data, mainly because of new algorithms and the increase of computational power. The power of the Bayesian Networks to represent in a compact and visually easy way the joint probability of a set of variables, make it ideal candidate for the complex data of a communication network.

In this thesis work, most of the focus will be put on learning an acceptable structure by a combination of expert's knowledge and appropriate learning methods. Two main approaches were investigated by using a Hybrid Network(mix of continuous and discrete data) and a full discretization of the variables. Although both networks managed to capture expected associations between the variables, the hybrid network poses serious restrictions that are not well supported by the data.

In order to investigate more approaches to structure learning, the Hill Climbing algorithm was enhanced with random restarts. As a third option the bootstrap method was tested also as a different way to construct more robust models and estimate the strength confidence of both the arcs and their direction. Finally a comparison of the predictive power for the different learning techniques was evaluated through cross validation.

Acknowledgements

I would like to thank my supervisors Erik Eklund ,David Andersson from Ericsson and Professor Larisa Beilina from the Chalmers university of Technology for their support and guidance during he completion of the thesis.

Themistoklis Christos Mouliakos, Gothenburg, December 2019

Contents

List of Figures **xi**

List of Tables **xiii**

1 Theoretical Background **5**
1.1 Some notions and definitions of Probability theory 5
1.2 Notions of Graph Theory 7
1.3 Conditional Independence and DAGS 9
1.3.1 D-Separation 12
1.4 Markov equivalence and PDAGs 14
1.5 Discretization of continuous variables 15
1.6 Conditional Linear Gaussian Bayesian Networks 17
1.7 Causality in DAGs 18
1.8 Summary 19

2 Bayesian Networks : Learning and Inference **21**
2.1 Structure learning 21
2.1.1 Constraint based algorithms 22
2.1.2 Score based algorithms 26
2.2 Scoring Functions 27
2.3 Maximum Likelihood Parameter learning 29
2.4 Inference 30
2.4.1 Logic Sampling 30
2.5 Bootstrap and Model Averaging 32
2.6 Summary 34

3 Computational Results **35**
3.1 Domain Insights 35
3.2 Hybrid/Gaussian Model Results 36
3.3 Discrete Model 38
3.3.1 Results 39
3.4 Discussion 41

A Data Plots **III**

B Discrete Data Tables **IX**

C Hartemink Method **XI**

D Maximum likelihood **XV**

E Number of parameters **XVII**

List of Figures

1.1 Directed graph with four nodes. A is the root. B has A as parent. C and D are leafs because they have no children. A and B are ancestors of D. B and C are children of A. Nodes B and D are descendants of A. 8
1.2 A directed cycle. 9
1.3 A Directed acyclic graph. 9
1.4 Undirected graph. Nodes 3 and 1 define a clique but not a max clique since we can always add node 2 and still have a clique. The maximal cliques are $\{1, 2, 3\}$, $\{2, 3, 4\}$, $\{3, 5\}$. 10
1.5 Proposed structure for a hypothetical example. 10
1.6 Serial connection 11
1.7 Converging connection 11
1.8 Diverging connection 11
1.9 Example graph to illustrate the d-separation concept. 13
1.10 Markov blanket in DAGS. 14
1.11 Markov blanket in Markov graphs. 14
1.12 The first three graphs belong to the class $[(X \perp Z|Y), \neg(X \perp Z)|\emptyset]$, the fourth graph belongs to the class $[(X \perp Z|Y), \neg(X \perp Z)|\emptyset]$. The last two graphs illustrate the corresponding PDAGs. 15

2.1 Calculating the Markov Blanket for node A. The structure of the original BN is shown with dashed lines (source : [21]) 25
2.2 Example of three nodes representing the connections between three discrete variables. 32
2.3 Query run using Logic sampling with 10,000 trials on the left and 100,000 on the right. Each run consisted of 500 repetitions of the query. The counts in the histogram, sum up to 500. 32

3.1 Auto correlation plot for one of the KPIs before re sampling. Since the vertical lines are way above the limits set by the horizontal lines, suggests strong correlation between the consecutive data points. . . . 36
3.2 Auto correlation plot for one of the KPIs after re sampling. The auto correlation reduces over time due to random sampling of the data set. 36
3.3 Structure estimation from mixed data using HC algorithm starting from an empty initial graph. The model is a CLGN where arcs from KPIs to A variables and arcs between the KPIs, where blacklisted due to domain knowledge. 37

3.4 Structure estimation from mixed data using HC algorithm together with 10 random restarts, 20 perturbations on starting graphs on each restart and random initial graph. The red dashed lines are the false positive arcs and the blue dashed lines the false negatives. 38
3.5 Structure estimation from mixed data using HC with bootstrap with 200 replications and a pre-estimated threshold. The red dashed lines are the false positive arcs whereas the blue dashed lines the false negative arcs. 38
3.6 Structure learnt from data after discretization. The HC algorithm was used with random initial graph. The edges from KPIs to A variables were blacklisted together arcs between KPIs based on domain knowledge. 39
3.7 Structure learnt from discrete dataset. The HC algorithm used with 10 random restarts and 20 random permuted arcs after each restart. The red dashed lines correspond to the false positives arcs and with dashed blue lines the false negatives arcs. 40
3.8 Structure learnt from discrete dataset. The HC algorithm used with bootstrap approach with 200 number of replicates.The blue dashed lines correspond to the false negatives arcs. 40

A.1 Barplot of A variables. IV
A.2 Histogram of continuous variables before discretisation. IV
A.3 Histogram of continuous variables before discretisation. V
A.4 Discretised variables with quantile method. Ir creates bins of equal frequency of points. Does not preserve initial distribution of data. . . V
A.5 Discretised variables with interval method. It creates uneven distribution of points in bins and it resembles the initial distribution. . . . V
A.6 Time evolution of the KPI1. Different colours correspond to different levels. VI
A.7 Time evolution of the KPI2. Different colours correspond to different levels. VII

E.1 Toy dataset with mixed type of variables XVII

List of Tables

C.1 Toy data frame based on the iris data set. XI
C.2 The dataframe after the initial discretization using the interval method. The variables are binned into four levels each. XI
C.3 The dataframe after the after collapsing $[4.4, 4.65], (4.65, 4.9]$. XII
C.4 The dataframe after the after collapsing $(4.65, 4.9], (4.9, 5.15]$. XII
C.5 The dataframe after the after collapsing $(4.9, 5.15], (5.15, 5.4]$. XIII

E.1 Number of parameters learnt for each variable in the toy DAG. XVIII

§

Introduction

> I basically know of two principles for treating complicated systems in simple ways: the first is the principle of modularity and the second is the principle of abstraction. I am an apologist for computational probability in machine learning because I believe that probability theory implements these two principles in deep and intriguing ways — namely through factorization and through averaging. Exploiting these two mechanisms as fully as possible seems to me to be the way forward in machine learning
>
> Michael I. Jordan from [1]

Trying to model complex systems, involves great deal of uncertainty and using the power of probabilities is a great way to go [2]. Going this way though we need to deal with the joint distributions which can be both computationally and statistically expensive. With graphical models one tries to represent a joint distribution with a small number of edges allowing, in that way, to create models with reasonable both the computational cost and the number of parameters needed to estimate. The fundamental idea is to factorize the probability distribution in a way that the individual parts are easier to compute. There exist two great families for graphical models, *directed* and *undirected* ones, where the main difference from a computational point of view is the existence of a normalisation constant called *partition function*, for the undirected graphs.

Historically, probabilistic models in combination with graphs can be found at the early 1900's, for example, the *path diagrams* of the geneticist *Sewall Wright* [3]. Although the term Bayesian Networks can be traced back to **Judea Pearl** in 1985 and mainly because of **Thomas Bayes** (1702-1761) work on the way of updating probabilities in the light of new evidence. Soon Bayesian Networks became a popular tool in Artificial Intelligence and expert system's community for modelling uncertainty. Example of graphical models are **Markov chains**, **Naive Bayes**, **Neural Networks** etc.

Another term closely related in modelling joint distributions, is *generative models* and they have the great advantage that they need fewer number of parameters to learn than the size of training dataset. An example from deep learning in images is given in [4] where the authors trained a generative model to assign high and low probabilities on whether an image shows a bedroom. By sampling from this distribution, they created realistic images of bedrooms. Transferring the same principles, one can apply probabilistic models for *speech recognition* and *speech syn-*

thesis. Fascinating experiments with recurrent neural networks, like in [5], resulted in *text generation*. Probabilistic models were also applied in computational biology, as presented in [6]. In *medical diagnostics* we see the efforts in an older paper [7] to construct a Bayesian network to distinguish patients with pneumonia from other deceases, with very good results.

The thesis structure is as follows : **chapter one** will introduce a basic understanding of the Bayesian Networks. This includes definitions and terms from probability theory and graph theory together with topics related to Bayesian Networks and discretization techniques. In **chapter two** the topic of learning a Bayesian Network will be illustrated that consists of the **structure learning** and **parameter learning**. The **inference** phase is also discussed. There will be a small presentation of the algorithms for structure learning with most attention to score based type and specifically the **Hill Climbing** search. Computational results, discussion and future work, will be the topic of **chapter three**.

1

Theoretical Background

A Bayesian network is a **Directed Acyclic Graph** which aims to represent in a compact way the joint probability distribution of a set of random variables. In situation where the random variables are all boolean, in order to learn all the entries of the conditional probability table is 2^N where N is the number of variables. This number becomes huge as N becomes large and quickly the task of calculating the empirical frequencies(or empirical probabilities) becomes intractable. In theory, if the joint probability is known, one can answer to any type of queries of interest. This task can be made more easy if we introduce/exploit any conditional Independence between the variables which will reduce the number of probabilities to learn.

This Chapter introduces background knowledge in probability and graph theory in order to support more advanced notions needed for introducing Bayesian Networks. Because the thesis is going to investigate both discrete and continuous networks, transformation of continuous variables to discrete is needed, thus there will be also a presentation of discretization methods of continuous models.

1.1 Some notions and definitions of Probability theory

Now is going to follow a brief presentation of some notions which will be used through the thesis.

Definition 1.1. Discrete variables case rule. The probability $P(X = x)$ of a variable X to be in the state x, is represented by a value between 0 and 1. If $P(X = x) = 1$ then it is certain for the random variable to be in that state and if $P(X = x) = 0$ it is certain that the variable is not in that state. If $dom(X)$ the domain of the random variable X, then $\sum_{x \in dom(X)} P(X = x) = 1$, i.e the sum of the probabilities over the space of all states is 1.

Definition 1.2. Marginals. If we have the joint probability distribution of two random variables $P(X = x, Y = y)$ then the distribution of one of them is given by :

$$P(X = x) = \sum_i P(X = x, Y = y_i)$$

here, for the interaction of two variables we have the following form :

$$P(\text{X=a or Y=b}) = P(X = a) + P(Y = b) - P(\text{X=a and Y=b})$$

where $P(\text{X and Y}) = P(X, Y)$.

Definition 1.3. Conditional Probability and Baye's rule. The probability of X occurring given that Y is true :

$$P(X|Y) = \frac{P(X, Y)}{P(Y)} \tag{1.1}$$

for $P(Y) \neq 0$. Using that $P(X, Y) = P(X, Y)$, we arrive at Baye's rule :

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)} \tag{1.2}$$

Bayes theorem is in the center of probabilistic reasoning i.e. detecting conditional probability relationships from data. Looking at eq. (1.2) the term $P(X|Y)$ is the posterior or conditional probability of observing X given we have observed Y. The term $P(Y|X)$ is called the likelihood for a given $x \in X$ and it is the probability of observing Y given the evidence on X and $P(X)$ is the probability distribution of X, known as the prior. Finally the term $P(Y)$ is the probability distribution of the evidence of Y and acts as a normalization constant. If we cannot compute $P(Y)$, then we get the unnormalized posterior.

Definition 1.4. Probability density function. In the case of a continuous random variable x, the probability density $f(x)$ is defined such that :

$$\int_{-\infty}^{+\infty} f(x)dx = 1, \quad f(x) \geq 0$$

If instead someone wants the probability of the random variable inside a specific interval $[A, B]$, then it is computed as :

$$P(A \leq x \leq B) = \int_{A}^{B} f(x)dx.$$

Definition 1.5. Independence. Mathematically we say that two random variables, X and Y, are independent if it holds :

$$P(X, Y) = P(X)P(Y), \tag{1.3}$$

which is equivalent also to say :

$$p(X|Y) = p(X) \quad \Leftrightarrow \quad p(Y|X) = p(Y). \tag{1.4}$$

if eqs. (1.3) to (1.4) are valid for all states of X and Y, then they are independent.

The notions of independence and conditional independence play an import role in Bayesian networks. Despite their mathematical formulation, human reasoning can understand those concepts without using any mathematics [8] or a need to perform numerical calculations. It is more of a common sense to argue that the probability of a nuclear war happening in the next years is independent of winning the lottery without having to verify something like $P(X,Y) = P(X)P(Y)$. An event A is said to be independent of event B, for a given K and stated as :

$$P(A|B,K) = P(A|K)$$

The notion of **conditional independence** give us the the intuition in how dependencies change in the light of new information. Since K is given , knowing B will not change the probability of A. Later in the thesis it is shown how those concepts relate to Bayesian networks and probabilistic graphical models in general. Generalizing eq. (1.1) and using the *chain rule*, we have the following expression

$$P(X_1 \dots X_n) = \prod_{i=n}^{1} P(X_i|P(X_{i-1} \dots P(X_1).$$

1.2 Notions of Graph Theory

Now it follows a brief presentation of some definitions from graph theory.

Definition 1.6. A graph G consists of set of nodes, $\mathcal{X} = \{X_1 \dots X_n\}$ and a set of edges $\mathcal{E} = \{(X_i, X_j) : X_i, X_j \in \mathcal{X}\}$, connecting those nodes and it is defined as $G = G(\mathcal{X}, \mathcal{E})$. An edge connecting two nodes, X_i, X_j, can have a direction i.e. $X_i \rightarrow X_j$ or not, $X_i - X_j$. If all edges in the graph fall into the first case we have a **directed graph** whereas if they fall in the second case we have an **undirected graph**.

Definition 1.7. A graph can be represented by the so called **adjacency matrix** A, which is a square matrix of the size $n \times n$, where n is the number of nodes. The entries of the matrix are usually, 0's and 1's. If we have the edge $X_i \rightarrow X_j$ then $A_{ij} = 1$, but if there is no connection between X_i, X_j then $A_{ij} = 0$. In the case of an *undirected* graph the table is **symmetric**.

Definition 1.8. If the following holds

$$G(X_i, X_j) = 1 \quad \textbf{IFF} \quad G(X_j, X_i) = 1,$$

then the graph is undirected. If now

$$G(X_i, X_i) = 0.$$

holds, there are no self loops.

Definition 1.9. For a node X_j, in a *directed graph*, the set

$$pa(X_j) \doteq \{X_i : G(X_i, X_j) = 1\},$$

is called **parents** of X_j. Further, for the same node, the set

$$ch(X_j) \doteq \{X_j : G(X_j, X_1) = 1\},$$

is called **children** of X_j. For a node X_j, its **family** is the node and its parents i.e.

$$fam(X_j) = \{X_j\} \cup pa(X_j).$$

A node without parents is called **root** and a node without children is called **leaf**. We call all the parents, grandparents and so on, **ancestors** of a node. We call all the children, grandchildren and so on, **descendants** of a node (See fig. 1.1).

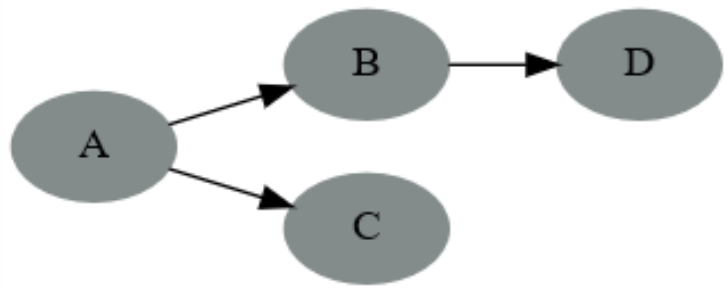


Figure 1.1: Directed graph with four nodes. A is the root. B has A as parent. C and D are leafs because they have no children. A and B are ancestors of D. B and C are children of A. Nodes B and D are descendants of A.

Definition 1.10. Path or trail $X_i \rightsquigarrow X_j$, is a series of edges from X_i to X_j. By **cycle** we define a series of nodes that if we follow them we get to the node we started, $X_1 - X_2 - \ldots X_n - X_1$ where $n \geq 2$. If the graph is directed then we have a **directed cycle**(See fig. 1.2).

Definition 1.11. A directed graph with no directed cycles is called **directed acyclic graph** or DAG. In fig. 1.1 we see such an example.

Definition 1.12. A topological ordering of a DAG is when the parent nodes have lower numbers than those of their children. For example in fig. 1.3 the topological orderings can be written as $\{1, 2, 3, 4\}$ or $\{1, 3, 2, 4\}$.

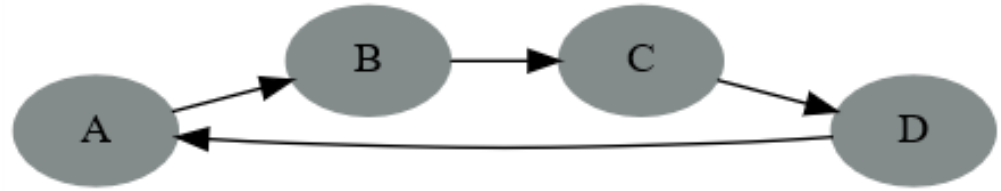


Figure 1.2: A directed cycle.

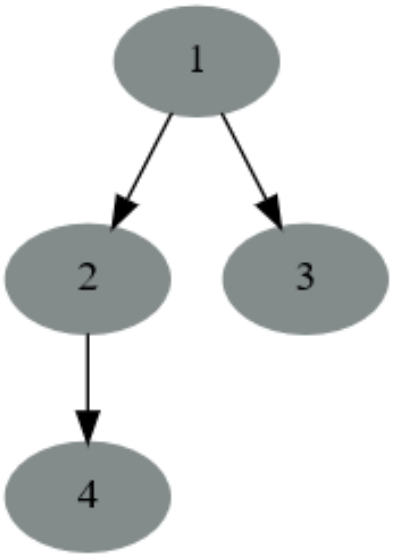


Figure 1.3: A Directed acyclic graph.

Definition 1.13. A **Directed tree** is a DAG where every node has at most one parent. **Undirected tree** is an undirected graph where there exists only a single path between any pair of nodes (no cycles).

Definition 1.14. By **neighbors** of node we mean any other node that is directly connected to it. This is true for any graph.

Definition 1.15. In the case of an *undirected graph*, a **clique** is the set of nodes that are all neighbors to each other. By **maximum clique** we call the clique that cannot be made any larger and remain a clique (See fig. 1.4).

1.3 Conditional Independence and DAGS

Assume that the structure in fig. 1.5 represents the connections of a set of variables for a hypothetical problem .

For constructing the network under consideration, the assumption that every variable is *independent of its non-descendants given its parents*is made. This is called

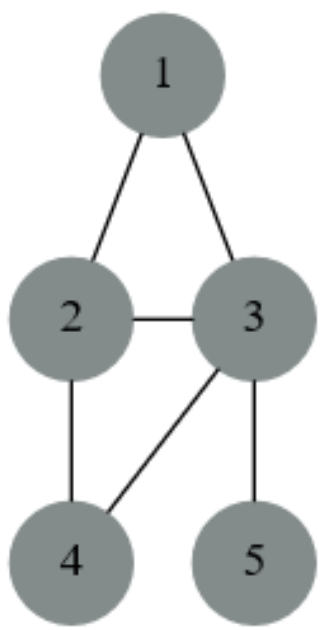


Figure 1.4: Undirected graph. Nodes 3 and 1 define a clique but not a max clique since we can always add node 2 and still have a clique. The maximal cliques are $\{1, 2, 3\}$, $\{2, 3, 4\}$, $\{3, 5\}$.

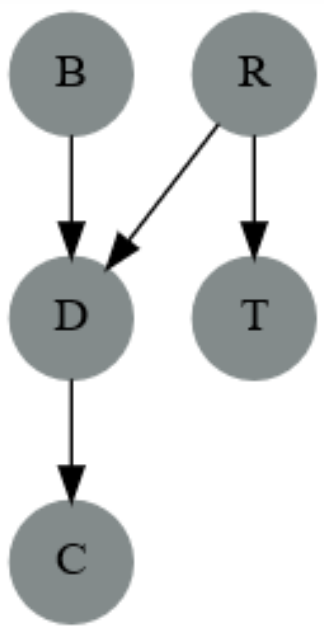


Figure 1.5: Proposed structure for a hypothetical example.

local Markov property and for in this example it states that if variable D is given, then B and R don't depend on T. In order to do perform inference we need to make queries on the joint distribution but without the need to use directly the full conditional probability table (for discrete variables). Bayesian networks have the benefit to read from this table with fewer calculations. To get some intuition and for small number of variables, we can list our variables in order such that there

are no non-descendants on the left : C,D,B,T,R. For the joint probability we have then :

$$\begin{aligned} P(C, D, B, T, R) &= P(C|D, B, T, R)P(D|B, T, R)P(B|T, R)P(T|R)P(R) \\ &= P(C|D)P(D|B, R)P(B)P(T|R)P(R) \end{aligned}$$

Thus every entry from joint probability table can be calculated since any entry is some combination of the above variables.
There are three types of connections in Bayesian networks : Serial (or head-tail), Converging (or head-head) and Diverging (tail-tail).

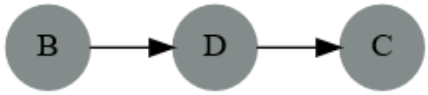


Figure 1.6: Serial connection

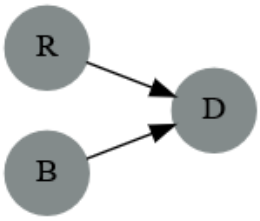


Figure 1.7: Converging connection

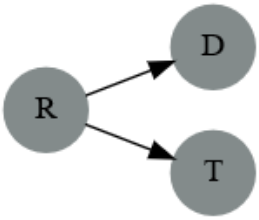


Figure 1.8: Diverging connection

In terms of conditional independence and for the serial connection, knowing D makes B and C independent (intermediate cause). In fig. 1.7 NOT knowing D or C makes R and B independent (common effect). For the last case in fig. 1.8, knowing R makes T and D independent (common cause). More specifically :

- For the **Serial** connection (fig. 1.6) the statement is knowing D makes B, C conditionally independent, $P(B, C|D) = P(B|D)P(C|D)$:

$$\begin{aligned} P(B, C|D) &= \frac{P(B, C, D)}{P(D)} \\ &= \frac{P(B)P(D|B)P(C|D)}{P(D)} \\ &= P(B|D)P(C|D) \end{aligned}$$

  the last equality is because :

$$\begin{aligned} P(D|B)P(B) &= P(B, D) \\ &= P(D|B)P(D). \end{aligned}$$

  So B and C are conditionally independent given D.

- For the **Diverging** connection (fig. 1.8) the statement is knowing R makes T and D independent :

$$P(R, D, T) = P(R)P(D|R)P(T|R) \tag{1.5}$$

Conditioning on R it holds that

$$\begin{aligned} P(D, T|R) &= P(R, D, T)/P(R) \\ &= P(D|R)P(T|R) \end{aligned}$$

So here also D and T are conditionally independent given R.

- For the **Converging** connection (fig. 1.7)the statement is NOT knowing D or C makes R and B independent :

$$P(R, D, B) = P(R)P(B)P(D|R, B)$$

but now

$$\begin{aligned} P(R, B|D) &= P(R, B, D)/P(D) \\ &= P(R|D)P(B|D) \end{aligned}$$

is not true in general. Instead now R and B are *marginally independent* because

$$\begin{aligned} P(R, B) &= \sum_D P(R, B, D) \\ &= P(R)P(B)\sum_D P(D|R, B) \\ &= P(R)P(D) \end{aligned}$$

1.3.1 D-Separation

The concept of D-separation is very important for studying directed acyclic graphical models and is ,in a sense, a generalisation of the conditional independence relations that were presented above. The letter *d* in d-separation stands for *dependence* according to some authors[1]. Following [8] we have the following definition :

Definition 1.16. If **X**,**Y**,**Z** are three disjoint subsets of nodes of a DAG. We will say that **Z d-separates X** from **Y** if along every path between a node in **X** and a node in **Y** there is a node **w** satisfying one of the following conditions :

1. **w** has converging arrows and none of **w** and its descendants are in **Z**, **or**
2. **w** does not have converging arrows and **w** is in **Z**.

If a path satisfies any of the above, is said to be **active** and when it does not satisfies any of the above, is said to be **blocked** by **Z**. Checking the example in fig. 1.9, if

[1] https://www.andrew.cmu.edu/user/scheines/tutor/d-sep.html

$\mathbf{Z} = \{1\}$ and $\mathbf{Y} = \{3\}$, $\mathbf{X} = \{2\}$ then the path $2 \leftarrow 1 \rightarrow 3$ is said to be blocked because the node $1 \in \mathbf{Z}$ satisfies the second condition. Thus $\mathbf{Z} = \{1\}$ **d-separates** $\mathbf{X}$ and $\mathbf{Y}$. Further, the path $2 \rightarrow 4 \leftarrow 3$ is blocked because node 4 satisfies condition 1. But if instead $\mathbf{Z} = \{1, 5\}$ then the path $2 \rightarrow 4 \leftarrow 3$ becomes active.

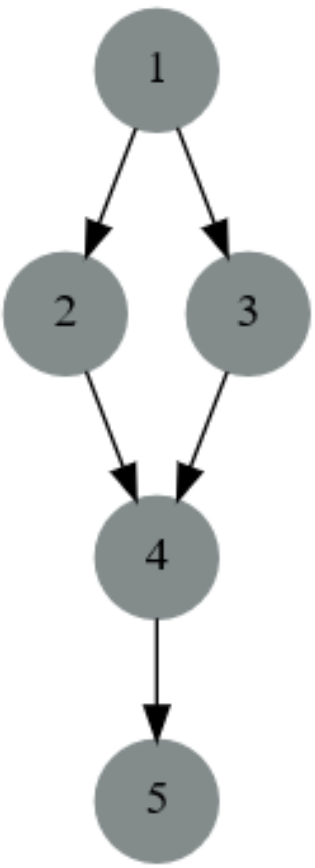


Figure 1.9: Example graph to illustrate the d-separation concept.

Intuitively, the statement **Z d-separates X** from **Y** says that X and Y are conditionally independent given the variable Z in all the probability distributions that the specific graph can represent. Further an active path allows information/dependence to flow between the variables of the two nodes. If there is any active path between two nodes, the it is said that they are **d-connected**. A very efficient and popular algorithm to test if **Z d-separates X** from **Y**, is the **Bayes ball algorithm** as described in [9].

Assume the graph $\mathcal{G}$ and $\mathcal{I}(\mathcal{G})$ is the set of all conditional statements encoded in $\mathcal{G}$. It is said that $\mathcal{G}$ is an **I-map** for P iff $\mathcal{I}(\mathcal{G}) \subseteq \mathcal{I}(\mathcal{P})$, where $\mathcal{I}(\mathcal{P})$ the set of all conditional statements that hold for the distribution P. In words, the previous statement means that the graph is a **I-map** only if the conditional independence statements made, hold true for the distribution also. The converse is not true : a distribution may factorize over $\mathcal{G}$ but there are independencies not captured by $\mathcal{G}$. In the special case when no conditional independence assumptions are made, the

graph is fully connected and is a *I-map* for all distributions. It is said that $\mathcal{G}$ is a **minimal I-map** of the distribution P, if $\mathcal{G}$ is a *I-map* of P and there exists no $\mathcal{G}' \subseteq \mathcal{G}$ which is a *I-map* of P. A relevant definition to I-map is the **D-map** : a graph $\mathcal{G}$ is a *D-map* of the distribution P if every conditional independence statement in P holds true for G. If G is both a D and a I map for P, then it is called **perfect map**.

Definition 1.17. Markov blanket is the minimal set of nodes for a node X_i that if observed, X_i becomes independent of the rest of the nodes in the graph. The notion is common for both directed and undirected graphs. In undirected graphs the Markov blanket is just the neighbouring nodes of X_i whereas in DAGs it is the union of parents, children and the co-parents(other parents of its children) of X_i. In fig. 1.10 and fig. 1.11 the Markov blanket of node three is shown in dark grey colour.

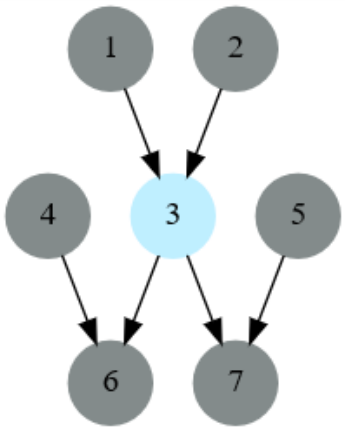


Figure 1.10: Markov blanket in DAGS.

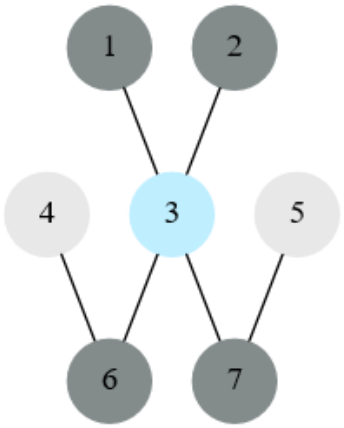


Figure 1.11: Markov blanket in Markov graphs.

1.4 Markov equivalence and PDAGs

Despite any differences in their structure, two DAGs can encode the same conditional independencies and are equivalent in this way. This concept was studied and presented by *TS Verma* and *Judea Perl* in [10] giving the formal definition that two DAGs are equivalent iff they have the same **skeleton** and the same **v-structure**. The skeleton is the graph when the direction from the edges is removed and v-structure are three nodes with two edges starting from two disjoint nodes, converge to the third (the head to head connection show in fig. 1.7). The equivalence between DAGs plays an important role since it is involved heavily in algorithms that try to learn the structure of a Bayesian network from data and a lot of research is done in that area, like in [11].
A set of Markov equivalence classes can be represented uniquely by a PDAG(partially DAG) which is a graph that contains both directed and udirected edges. By assigning directions to the undirected edges from a given PDAG, with the constraints of not making any directed cycles and no new v-structures, one can extract Markov equivalent DAGs. For example, the skeleton X-Y-Z, has two equivalent classes :

one when $[(X \perp Z|Y), \neg(X \perp Z)|\emptyset]$ and $[(X \perp Z|Y), \neg(X \perp Z)|\emptyset]$ (see fig. 1.12) (example taken from [12]).

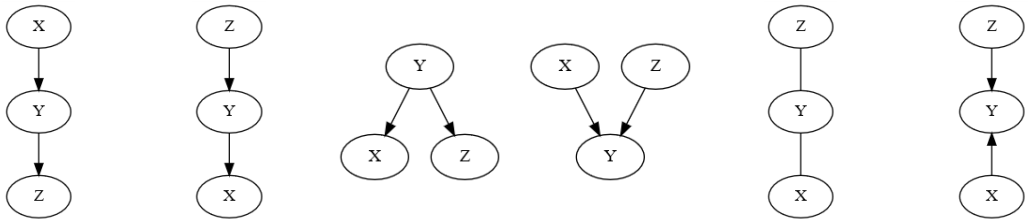


Figure 1.12: The first three graphs belong to the class $[(X \perp Z|Y), \neg(X \perp Z)|\emptyset]$, the fourth graph belongs to the class $[(X \perp Z|Y), \neg(X \perp Z)|\emptyset]$. The last two graphs illustrate the corresponding PDAGs.

One important observation is that diverging and serial connections have equivalent factorizations. This means that one can lead to the other by applying the Baye's rule. This kind of equivalence is characteristic to Markov equivalent structures and the set of all equivalent structures is called equivalent class.

1.5 Discretization of continuous variables

Lot of the research and applications of Bayesian networks deals with cases where either the variables are discrete or they discretize the continuous variables. This of course comes at a cost of losing information during the process of binning the data and also increases the number of parameters that have to be learned because more levels for a variable are produced. A very simple and easy to implement method, is the **interval** approach. It only breaks the variable into equally width levels . Some of the drawbacks of this method are : sensitivity to outliers, there might be intervals with no points at all, it does not take into account the mutual information inside the data set. Also it does not modify the shape of the distribution. A second method is the **Quantile** which tries to create levels of equal size. It is also fast to implement and it is not sensitive to outliers. Quantile method balances the shape of the distribution and keeps more information than the interval method. In both cases the number of intervals to create, is not known. Some approaches are: *Brooks-Carruthers*, *Huntsberger*, *Sturges*, *Scott*, *Freedman-Diaconis* [2].

A more sophisticated approach that tries to preserve the mutual information between the variables, is the method described by Hartemink in [13]. Next we describe the Hartemink discretization algorithm , as presented in [14].
The algorithm 1 starts with an initial discretization of variables into k_1 intervals. Then it iterates inside the variables and looks at the adjacent intervals and collapses the ones that minimise the loss of information between the intervals. The algorithm stops when $k_2 << k_1$. The Hartemink approach is a better at keeping information compared to quantile and interval methods. The method does not take into account more complex relationships and there is still information loss.

[2] http://eric.univ-lyon2.fr/ ricco/cours/slides/en/discretisation.pdf

Algorithm 1: Hatermink Discretization algorithm.

1. Discretize each variable independently using an initial discretization method like quantile, using a large number of intervals k_1 like eg. 50 or 100.

2. Repeat the following steps until each variable has $k_2 << k_1$ intervals, iterating over each variable X_i, $i = 1 \ldots, p$:
   (a) Compute :
   $$M_{X_i} = \sum_{j \neq i} MI(X_i, X_j) \tag{1.6}$$
   (b) For each pair of adjacent intervals l of X_i, join them into one interval and with the resulting variable $X_i^*(l)$ compute :
   $$M_{X_i^*(l)} = \sum_{j \neq i} MI(X_i^*(l), X_j) \tag{1.7}$$
   (c) set $X_i = argmax_{X_i(l)} M_{X_i^*(l)}$.

The mutual information between two discrete variables $\mathbf{X}$ and $\mathbf{Y}$, gives a measure of the dependence or the amount of information for one random variable when observing the other one. Given a data set of N observations, were the columns are the variables, the mutual information is calculated though the empirical probabilities :

$$MI(X, Y) = \sum_{i}^{N} \sum_{j}^{N} p(x_i, y_j) \log \left(\frac{p(x_i, y_j)}{p(x_i)p(y_j)} \right) \tag{1.8}$$

Another way to define the mutual information is through the entropy :

$$H(X) = -\sum_{i}^{N} p(x_i) log(p(x_i)), \tag{1.9}$$

and for the two random, variables case :

$$H(X, Y) = -\sum_{i}^{N} \sum_{j}^{N} p(x_i, y_j) log(p(x_i, y_j)). \tag{1.10}$$

In terms of the entropy, the mutual information of two discrete random variables, is given as :

$$MI = H(X) + H(Y) - H(X, Y).$$

Sometimes it is very common to use the log_2 instead of the log in eqs. (1.8) to (1.10) and then the results are measured in units of bits.
The mutual information is linked also to the $\mathbf{G-test}$ which is a likelihood ratio test, by :

$$G = 2NMI(X, Y) \tag{1.11}$$

where N is the total observations of the dataset. It is often the case that many software libraries use eq. (1.11) as a scaled version of the mutual information score. A toy example for the method is shown in appendix C.

1.6 Conditional Linear Gaussian Bayesian Networks

Conditional Linear Gaussian Bayesian Networks (CLGN) are a popular class of Bayesian networks for applying in the case of mixed variables type. They have the restriction that discrete variables cannot have continuous parents. Those limitations arise from the fact that the joint probability distribution must factorise in a discrete part and a mixed part. The joint CPD of a CGN is a mixture of Gaussian distributions. There exists a Gaussian distribution for each level of the discrete variables. The weight of each Gaussian in the mixture is the probability of this level [15].

The discrete nodes have conditional probability tables as their local distributions and the continuous variables have a linear regression model for each level of their discrete parent(s). Let Δ_{X_i} be the discrete parents of the node X_i and Γ_{X_i} be the continuous parents. Then we can express the local distributions as :

$$X_i|\Pi_{X_i} \sim \mathcal{N}(\mu_{X_i,\delta_{X_i}} + \Gamma_{X_i}\beta_{X_i,\delta_{X_i}}, \sigma_{X_i,\delta_{X_i}}) \tag{1.12}$$

where for the linear regressions we have :

$$X_i = \mu_{X_i,\delta_{X_i}} + \Gamma_{X_i}\beta_{X_i,\delta_{X_i}} + \epsilon_{X_i,\delta_{X_i}} \qquad \epsilon_{X_i,\delta_{X_i}} \sim \mathcal{N}(0, \sigma_{X_i,\delta_{X_i}}) \tag{1.13}$$

where δ_{X_i} is the levels of the discrete variables. If a node has no discrete parents then its local distribution is given by the linear regression :

$$X_i = \mu_{X_i} + \Gamma_{X_i}\beta_{X_i} + \epsilon_{X_i} \qquad \epsilon_{X_i} \sim \mathcal{N}(0, \sigma_{X_i}) \tag{1.14}$$

Two main restrictions of this models are that discrete nodes in the DAG cannot have continuous parents and that they allow linear dependencies between the variables because of the linear regression relationships.

Below, in eqs. (1.15) to (1.20), the number of parameters and the log-likelihoods are shown for the local distributions in the cases of Bayesian networks.
The log-likelihood and the parameters for a conditional linear Gaussian network are given by :

$$LL(X_i, \Pi_{X_i}) = \prod_{m=1}^{n} \mathcal{N}(x_m; \mu_{X_i,\delta_m} + \gamma_m\beta_{X_i,\delta_m}, \sigma^2_{X_i,\delta_m}) \tag{1.15}$$

$$|\Theta_{X_i}| = |\Delta_{X_i}| \times (|\Gamma_{X_i}| + 1) \tag{1.16}$$

The log-likelihood and the parameters for a Gaussian Bayesian network are given by :

$$LL(X_i, \Pi_{X_i}) = \prod_{m=1}^{n} \mathcal{N}(x_m; \mu_{X_i} + \gamma_m\beta_{X_i}, \sigma^2_{X_i}] \tag{1.17}$$

$$|\Theta_{X_i}| = |\Pi_{X_i}| + 1 \tag{1.18}$$

and in the case of a discrete Bayesian network :

$$LL(X_i, \Pi_{X_i}) = \prod_{m=1}^{n} \mathbf{P}(X_i = x_m | Pi_i = \pi_m), \tag{1.19}$$

$$|\Theta_{X_i}| = R \times |\Pi_{X_i}| \tag{1.20}$$

In appendix E a simple example is shown, on how to calculate the number of parameters in CLGN.

1.7 Causality in DAGs

Although in the literature one can find interpretations of the arrows between two nodes as causal relationships or sometimes direct dependencies, caution is advised. The arrows don not necessarily describe causal relationships. As an example, take the connections : $B- > A- > C$, $B < -A- > C$ and $C- > A- > B$. In probabilistic terms those graphs are equivalent because they encode the same conditional independent statements :

$$P(A)P(B|A)P(C|A) = P(B)P(A|B)P(C|A)$$

describing the first graph whereas

$$P(A)P(B|A)P(C|A) = P(C)P(A|C)P(B|A)$$

describes the last graph.

In order to make causal interpretation of the graph, we need three assumptions :

1. Every node variable X_i is conditionally independent of its non effects given its causes. This is called **causal Markov condition**.

2. The DAG must be **faithful**, meaning that it is a perfect map between the probability distribution and the graphical representation.

3. The are no latent variables that may affect the structure of the DAG and act as confounding factors.

These assumptions are hard to verify in real life and possibly the best one can do is to check for confounding factor using blocking experimental design[3].

[3] http://www.bnlearn.com/about/teaching/slides-bnshort.pdf

1.8 Summary

Bayesian Networks are Directed Acyclic Graphs and model the joint probability distribution of a set of variables in a compact way. Bayesian Networks can be used as causal models but this requires more assumptions and especially the presence or not of latent variables. Id we want to discretise the continuous variables, then the Hartemink method is a good method to apply which tries to preserve the mutual information between the variables compared to more simple approaches. In the case we have a mixed type of variables, then the popular choice is the conditional linear gaussian networks. Although it has strong assumptions which may not be a good choice in a realistic situation.

2

Bayesian Networks : Learning and Inference

This chapter presents a brief review of the concept of learning a Bayesian network. There are two main phases of learning a Bayesian network : 1) Structure learning and the 2) Parameters learning. When the structure and parameters are known, one then can perform inference, which is the task of asking queries from our model. Although there is a number of structure learning algorithms, the focus was around score based type algorithms and particularly the *Hill Climbing*. The choice was based mostly for the fast approach and ease of interpretation. A brief review of most known structure learning algorithms will be presented also . For the parameter learning phase , the *Maximum Likelihood* method was used. Finally, there will be an illustration of *Logic Sampling* algorithm for approximate inference.

2.1 Structure learning

Structure learning is the first step in the process of learning a Bayesian network. This the stage of estimating a DAG from data. Alternatively structure can be estimated using experts knowledge at least in the case of relatively low number of variables. Learning a DAG from data is not an easy task and one of the reasons is that the space of possible structures becomes really huge with just a few variables only. Actually it is shown that the problem of structure learning is NP-Complete (see [16]) . The space of possible DAGs is super exponential on the number of nodes n ($\mathcal{O}(n!2^{n(n-1)/2})$) (see [17]). For example, for five variables there are about 10^4 DAGs whereas for ten variables, the number goes to about 10^{18}. Further, you can not find a unique structure for a given dataset, since many structures can encode the same conditional independence relations [18].

Compactly we can represent the problem of learning a model of a Bayesian network from a dataset $\mathcal{D}$ as :

$$\overbrace{P(\mathcal{M}|\mathcal{D})}^{\text{learning}} = \overbrace{P(\mathcal{G}|\mathcal{D})}^{\text{structure learning}} \times \overbrace{P(\Theta|\mathcal{G},\mathcal{D})}^{\text{parameter learning}} \tag{2.1}$$

were $\mathcal{G} = (V, E)$ is the structure of the Bayesian network that consists of the set of nodes V and the set of edges E and Θ are the parameters of the global distribution. Since the whole task of learning can be intractable, breaking into local computations the problem becomes feasible. Local computations are performed in the local

distributions involving one node X_i by decompose the global distribution $\mathbf{X}$. In terms of the local distributions the problem of a learning a Bayesian network can be formulated as :

$$P(\mathcal{G}|\mathcal{D}) \propto P(\mathcal{G})P(\mathcal{D}|\mathcal{G}) \tag{2.2}$$

and when using marginalisation :

$$P(\mathcal{G})P(\mathcal{D}|\mathcal{G}) = P(\mathcal{G}) \int P(\mathcal{D}|\mathcal{G},\Theta)P(\Theta|\mathcal{G})d\Theta \tag{2.3}$$

By denoting Π_{X_i} the parents of node X_i, $P(\mathcal{D}|\mathcal{G})$ can be written as :

$$P(\mathcal{D}|\mathcal{G}) = \prod_{i=1}^{N} \Big(\int P(X_i|\Pi_{X_i},\Theta_{X_i})P(\Theta) \Big) \tag{2.4}$$

and for the parameter learning :

$$P(\Theta|\mathcal{G},\mathcal{D}) = \prod_{i=1}^{N} P(\Theta_{X_i}|\Pi_{X_i},\mathcal{D}) \tag{2.5}$$

were the index i denotes the variables. There are mainly three major groups of algorithms for structure learning : **constraint**-based , **score**-based and **hybrid** algorithms.

2.1.1 Constraint based algorithms

Algorithms that fall into this category, originate from the work of *Verma and Perl*. In their work [10] Perl introduced the *Inductive Causation Algorithm* which aims for structure learning using statistical tests for conditional Independence. In algorithm 2 we can see the details of IC.

Algorithm 2: Inductive Causation Algorithm[19]

1. For each pair of variables A and B in V search for set $S_{AB} \subseteq V$ such that A and B are independent given S_{AB} and $A, B \notin S_{AB}$. If no such a set exists, place an undirected arc between A and B.

2. For each pair of non-adjacent variables A and B with a common neighbour C, check whether $C \in S_{AB}$. If it is not, set the direction of the arcs **A-C** and **C-B** to **A->C** and **C<-B**

3. Set the direction of arcs which are still undirected by applying recursively the following two rules:
   (a) if A is adjacent to B and there is a strictly directed path from A to B then set the direction of $A - B$ to $A- > B$.
   (b) if A and B are not adjacent but $A- > C$ and $C - B$, then change the latter to $C- > B$.

4. Return the resulting (partially) directed acyclic graph.

In the first step, finds all the pair of variables that are connected by an arc without caring for direction. Since those variables they cannot be d-separated they cannot be conditionally independent. In the second step it tries to find all the v-structures among non-adjacent variables A,B which have a common neighbour C. If there is a node C that d-separates A,B then it means that they are all part of the same v-structure with C as the center. By the end of this step both the skeleton and the v-structures are learned thus the equivalent class is uniquely identified. The last steps are all calculating the CPDAG(completed partially DAG). There is a problem with this is algorithm that in real life applications it is difficult to be applied since the first steps involve computations of conditional Independence relations which are exponential in numbers. Later, many algorithms where proposed and are computationally more efficient.

The **PC algorithm** was introduced in [20] and it is the first try for real life application of IC algorithm. In 2003 the **Grow-Shrink** algorithm was introduced [21]. In comparison with the PC algorithm, the Grow-Shrink(and others) make use of the Markov Blanket.

Algorithm 3: Grow-Shrink Markov Blanket algorithm.

1. $\mathcal{S} < -\emptyset$
2. While $\exists \mathbf{Y} \in \mathbf{U} - \{\mathbf{X}\}$ such that $\mathbf{Y}$,$\mathbf{X}$ are not independent given $\mathbf{S}$, do $\mathbf{S} < -\mathbf{S} \cup \mathbf{Y}$. [**Growing Phase**]
3. While $\mathbf{Y} \in \mathbf{S}$ such that $\mathbf{Y}, \mathbf{X}$ conditionally independent

In its core, the Grow-Shrink algorithm consists of two phases : growing and shrinking. As we can see in algorithm 3, it starts with an empty set S; As long as the variables examined, are dependent given the current state of S, it adds them to S. In the shrinking phase, the variables that violate the Markov Blanket property are deleted from S. In this way it identifies the Markov blanket of the node and further the direct parents and children of the node inside the blanket in order to create the local structure around the node. Using the d-separation criterion upon triples of variables, it then finds the direction of the edges. The GS Markov Blanket algorithm is of order $\mathcal{O}(n)$ where n is the number of independence tests. In fig. 2.1, an example of Grow-Shrink Markov Blanket algorithm is shown.

Since the Markov Blanket is recovered, this information is utilised to estimate the local structure for each node and thus estimating the whole structure of the Bayesian network. The plain Grow-Shrink then is presented in algorithm 4.

If $\max_X \mathbf{B}(X)$ bounded by a constant, then the algorithm is of order $\mathcal{O}(n^2)$ in conditional Independence tests.

Next the **Incremental Association Markov blanket algorithm** is presented as introduced in [22]. The algorithm consists of two phases : the forward selection

Algorithm 4: Grow-Shrink algorithm [21].

1. **Compute Markov Blanket** : For a node $\mathbf{X}$ compute $\mathbf{B}(X)$.

2. **Compute Graph Structure** : Let $\mathbf{T}$ the smaller in size set between $\mathbf{B}(X) - \{X\}$ and $\mathbf{B}(Y) - \{Y\}$ where $\mathbf{Y} \in \mathbf{B}(X)$. If $\mathbf{X}, \mathbf{Y}$ are dependent given $\mathbf{S}$ for all $\mathbf{S} \subseteq \mathbf{T}$, then $\mathbf{Y}$ is a direct neigbour of $\mathbf{X}$.

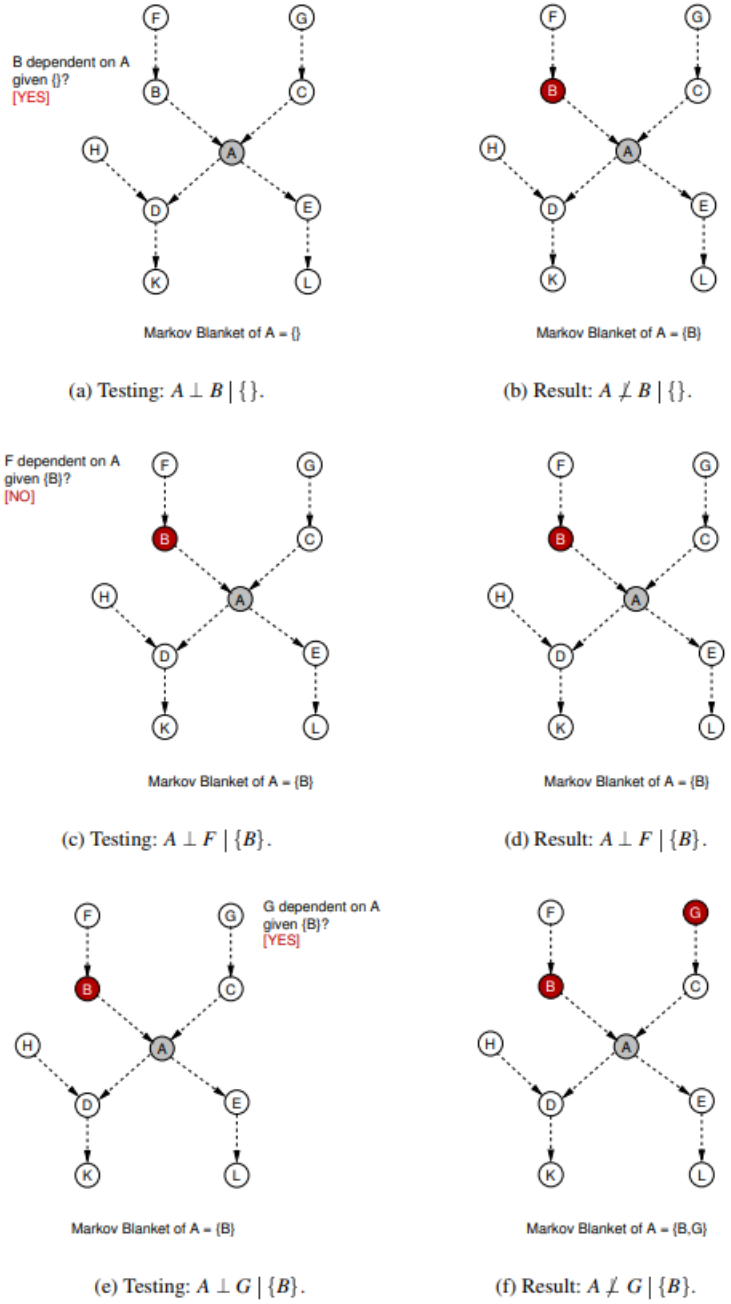
3. **Compute Edge Orientation** : $\mathbf{Y}$ is direct neighbor of $\mathbf{X}$. $\mathbf{T}$ is the smaller set between $\mathbf{B}(Y) - \{X, Z\}$ and $\mathbf{B}(Z) - \{X, Y\}$. If a variable $\mathbf{Z}$ exists such that $\mathbf{Z} \in \mathbf{N}(X) - \mathbf{N}(Y) - \{Y\}$($\mathbf{N}()$:denotes direct neighbor) and $\mathbf{Y}, \mathbf{Z}$ are dependent given $\mathbf{S} \cup \{\mathbf{X}\}$ for all $\mathbf{S} \subseteq \mathbf{T}$, then orient $\mathbf{Y} -> \mathbf{X}$.

4. **Remove Cycles** : As long as there are cycles in the graph do :
   (a) Compute the set of edges $\mathbf{C} : \{X -> Y$which is part of a cycle$\}$.

   (b) Remove from the current graph the edge in C that is part of the greatest number of cycles, and put it in $\mathbf{R}$.

5. **Reverse Edges** : Insert each edge from $\mathbf{R}$ in the graph in reverse order of removal in Step 4, reversed.

6. **Propagate Directions** : For all nodes and for all $\mathbf{Y} \in \mathbf{N}(X)$ such that neither $Y -> X$ nor $X -> Y$ execute the following rule until it no longer applies: If there exists a directed path from X to Y, orient $X -> Y$.



(a) Testing: $A \perp B \mid \{\}$.

(b) Result: $A \not\perp B \mid \{\}$.

(c) Testing: $A \perp F \mid \{B\}$.

(d) Result: $A \perp F \mid \{B\}$.

(e) Testing: $A \perp G \mid \{B\}$.

(f) Result: $A \not\perp G \mid \{B\}$.

Figure 2.1: Calculating the Markov Blanket for node A. The structure of the original BN is shown with dashed lines (source : [21])

and the backward one. Starting with the forward selection, a set **CMB** is defined that holds an estimate of the Markov Blanket $\mathbf{MB}(T)$. In the forward phase, **CMB** fills with variables of the $\mathbf{MB}(T)$ and false positives. The backward phase removes those false positives so that we end up in the end with the correct $\mathbf{MB}(T)$. The *Grow-Shrink algorithm* and the *Incremental Association algorithm* are very similar but the heuristics of the GS algorithm will produce more false positives in the CMB at the early stage of PHASE I (more on [22]).

Algorithm 5: Incremental Association algorithm [22].

```
[PHASE I] : CMB = ∅
while CMB has changed do
    Find the feature X in V − CMB − {T} that maximizes F(X; T|CMB);
    if not I(X; T|CMB) then
        add X to CMB;
    end
end
[PHASE II] : Remove from CMB all variables X for which I(X; T|CMB − {X})
 Return CMB
```

In 2005 the **Fast Incremental Association algorithm** was introduced by Yaramakala and Margaritis([23]) in order to calculate the Markov Blanket. Although this variant is very similar to other IAMB methods, it differs that it uses, not one but a number of attributes at a time after each re-ordering of the remaining attributes and thus reducing the amount of conditional independence tests needed. Also it uses $\mathbf{G}^2$ test which, by the opinion of the author, is more appropriate. The number of attributes added to the blanket at each iteration is based on the heuristic : add dependent variables as long as the conditional Independence tests are reliable(under the condition that there are enough data to conduct them). The authors claim that their algorithm gives faster and more reliable results at many occasions compared to other algorithms, without affecting the accuracy of calculating the Markov Blankets, in a bad way.

2.1.2 Score based algorithms

The idea behind this class of algorithms is to pick a candidate structure that maximizes a score function. An obvious limitation for these methods is the huge space of possible DAGs which is super exponential in the number of nodes. Most notable heuristics used are simulated anealling, greedy search, genetic algorithms (this topic is covered well in [24]). For our purposes we are going to present the Greedy Search algorithm as presented in [25] and uses a combination of Hill Climbing, TABU and random restarts steps.

Looking at algorithm 7, the first three steps are the usual Hill Climbing search which it can be improved by adding a Tabu search and random restarts. The random restarts phase is to assist HC not to get stuck in local minima by trying again the search from random initial states. The tabu search holds a list of previously visited states so that the algorithm will not visit them again. This contributes to both the efficiency and to avoid some local minima(see [24] for more).

Since looking to score all the possible arcs in the DAG space it will become unfeasible when the number of variables grow, some heuristics will be used. Hill Climbing will do a local search meaning that it will start from a initial graph structure and will add/remove/reverse one edge at a time, from the current best scored structure and

Algorithm 6: Fast Incremental Association Markov Blanket algorithm

$\mathbf{B}(T) \leftarrow \emptyset$
$\mathbf{S} \leftarrow \{A | A \in \mathcal{U} - \{T\}$ and$A \not\perp\!\!\!\perp T\}$
while $\mathbf{S} \neq \emptyset$ **do**
  $\langle X_1 \dots X_{|S|} \rangle \leftarrow \mathbf{S}$ sorted according to h
  insufficient data $\leftarrow$ FALSE
  [**GROWING PHASE**]
  for *i=1 to /S/* **do**
    if $(N/r_{X_i} r_T r_{B(T)}) \geq \kappa$ **then**
      $\mathbf{B}(T) \leftarrow \mathbf{B}(T) \cup \{X_i\}$
    else
      insufficient data $\leftarrow$ TRUE **go to**
    end
  end
  [**SHRINKING PHASE**]
  for *each attribute* $\mathcal{A} \in \mathbf{B}(T)$ **do**
    if $(\mathcal{A} \perp\!\!\!\perp T | \mathbf{B}(T) - \{\mathcal{A}\})$ **then**
      $\mathbf{B}(T) \leftarrow \mathbf{B}(T) - \{\mathcal{A}\}$
    end
  end
  if *insufficient data = TRUE* ***and*** *no attributes were removed in the shrinking phase* **then**
    halt
  else
    $\mathbf{S} \leftarrow \{\mathcal{A} | \mathcal{A} \in \mathcal{U} - \{T\} - \mathbf{B}(T)$ and$(\mathcal{A} \not\perp\!\!\!\perp T | \mathbf{B}(T))$
  end
end

score each configuration. Repeating this procedure, it will perform till the score can grow anymore. This means that at a local optima it will stop. Also since the space of DAGs grows super exponential with the number of variables,many local optima might exist and there is no guarantee that HC will find a global optimum. When enhancing HC with random restarts, we can select the number of random restarts and how perturbed the initial structure will be relative to the previous best estimate.

2.2 Scoring Functions

The main algorithm used for structure estimation is the Hill Climbing, which is a score based method. In order to perform model selection there few well known scores mainly : *Bayesian Information Criterion* (BIC) and *Akaike Information Criterion* (AIC).

BIC score is given by the formula :

Algorithm 7: Greedy Search algorithm using Hill Climbing and Tabu search together with a initialisation and random restart phases.

1. Compute the score of $\mathcal{G}$, $\mathcal{S}_{\mathcal{G}} = Score(\mathcal{G}, \mathcal{D})$.
2. Set $\mathcal{S}_{max} = \mathcal{S}_{\mathcal{G}}$ and $G_{max} = G$.
3. **Hill-Climbing :** repeat as long as S_{max} increases:
   (a) For every possible arc addition, deletion or reversal in G_{max} resulting in a DAG :
      i. Compute the score of the modified DAG G^*, $\mathcal{S}_{\mathcal{G}^*} = Score(\mathcal{G}^*, \mathcal{D})$
      ii. If $S_{G^*} > S_{max}$ and $S_{G^*} > S_G$ and $\mathcal{G}_{max} = \mathcal{G}$ and $S_G = S_{G^*}$

   (b) If $S_G > S_{max}$, set $S_{max} = S_G$ and $G_{max} = G$
4. **Tabu search**: for up to t_0 times :
   (a) repeat step 3 but choose the DAG $\mathcal{G}$ with the highest S_G that has not be visited in the last in the last t_1 steps regardless of S_{max};
   (b) if $S_G > S_{max}$ set $S_{max} = S_0 = S_G$ and $G_{max} = G_0 = G$ and restart the search from step 3.
5. **Random restart**: for up to r times, perturb G_{max} with multiple arc additions, deletions and reversals to obtain a new DAG G' and:
   (a) set $S_0 = S_{max} = S_G$ and $G_0 = G_{max} = G$ and restart the search from step 3.
   (b) if the new G_{max} is the same as the previous G_{max}, stop and return G_{max}.

$$BIC = ln(n)k - 2ln(L) \tag{2.6}$$

where n is the sample size, k the number of parameters learnt by the model and L the maximum likelihood of the data. BIC also has a great property that is very useful for scoring functions in Bayesian Networks that of being *decomposable* (i.e can break into local components- variables and their parents). The total score of a DAG will be the sum of BIC scores for each variable given its parents. The **AIC** is given by :

$$AIC = 2k - 2log(L) \tag{2.7}$$

Since the space of DAGs can be huge, in practice some heuristics is used together with a decomposable score. If $\mathcal{G}$ is the structure and Π_{X_i} the parents of the node X_i then :

$$Score(\mathcal{G}) = \sum_{i=1}^{N} Score(X_i|\Pi_{X_i}) \tag{2.8}$$

and then BIC is given by :

$$BIC(\mathcal{G}) = \sum_{i=1}^{N} logP(X_i|\Pi_{X_i}) - \frac{|\Theta_{X_i}|}{2} \log n \tag{2.9}$$

which is the version used in the programs tested in this thesis.

For Bayesian Networks with mixed variables a common choice for scoring function is the **Conditional Gaussian Score**(CG). CG score has two assumptions : 1) The data where generated from mixture of Gaussians, one for each level of the discrete variable and 2) The data are i.i.d because it helps with the computation of the log-likelihood. Also, CG score uses BIC at its core so it has also the property of being decomposable.

Let Y_i denote the variables in the Bayesian Network and Pa_i the corresponding parents. The parents are then divided into two disjoint sets of continuous and discrete type Pc_i and Pd_i. This method models three different sets : $(Y_i \cup Pa_i)$ when Y_i is continuous or discrete and the set of Pa_i. Despite some differences the procedure is almost the same if Y_i is continuous or discrete. In either case, initially the data must be partitioned to Π_i partitions for each combination of the discrete variable. Further a design matrix is constructed for each $p \in \Pi_i$ where it contains the continuous variables that are to be fitted using the Gaussian. The two last major steps are to fit also a multinomial distribution for counts and calculate the BIC score(more details in [26]). BIC score uses the following equations to calculate the score for the child given parent(s) system :

$$l_i(\hat{\boldsymbol{\theta}}|\mathbf{X}) = l_{\{Y_i \cup Pa_i\}}(\hat{\boldsymbol{\theta}}|\mathbf{X}) - l_{Pa_i}(\hat{\boldsymbol{\theta}}|\mathbf{X}) \tag{2.10}$$

for the log-likelihood and for the degrees of freedom :

$$df_i(\hat{\boldsymbol{\theta}}) = df_{\{Y_i \cup Pa_i\}}(\hat{\boldsymbol{\theta}}) - df_{Pa_i}(\hat{\boldsymbol{\theta}}) \tag{2.11}$$

2.3 Maximum Likelihood Parameter learning

After the structure is learnt, the next step is to fit the parameters. Let the index i goes through the different variables of the network, the index k goes through the data points, then :

$$L(\Theta : D) = \prod_k P(x_k : \Theta) \tag{2.12}$$

$$= \prod_k \prod_i P(x_{ik}|\Pi_{x_i,k} : \Theta_i) \tag{2.13}$$

$$= \prod_i \prod_k P(x_{ik}|\Pi_{x_i,k} : \Theta_i) \tag{2.14}$$

$$= \prod_i L_i(D : \Theta_i) \tag{2.15}$$

If one can assume that each local conditional probability distribution is independent from each other(disjoint), then the MLE can be computed by separately maximizing each local likelihood. In the case of table conditional probability distributions, this can be decomposed further and result in the following estimate :

$$\theta_x|p = \frac{\#\text{occurrences were} X = x|P = p}{\#\text{occurrences were} P = p} \tag{2.16}$$

a small example is demonstrated in eq. (D.1).

2.4 Inference

In inference we seek for the computation of marginal distributions of a subset of variables given some others and in general make predictions given new data. In general the most known classes of Bayesian networks that exact inference is possible, are the discrete, the conditional linear and the Gaussian ones ([27]). In the case of discrete Bayesian network, the number of parameters to learn grows exponentially with the number of variables like n^X, where n denotes the possible outcomes of the discrete variable. Famous examples of exact inference algorithms, are *variable elimination*, *message passing* and *junction tree*.

Since exact inference is impossible for big networks, approximate inference methods were studied that perform well in real time applications. This kind of inference is based on Monte Carlo methods. Below we present the **logic sampling** algorithm as proposed in [28] and discussed in [1]. These algorithms were used in the computations presented in section 3.

2.4.1 Logic Sampling

The problem can be formulated as follows : given a Bayesian network $(\mathcal{G}, \mathcal{P})$ where $\mathcal{G} = (\mathbf{V}, \mathbf{D})$ we want to calculate the probabilities of some values in a subset of some nodes given the evidence (observed values of an other subset of nodes). By $\mathbf{V}$ we denote the set of all nodes in the BN, $\mathcal{P}$ the joint probability distribution over the nodes and $\mathbf{D}$ the arcs in the BN. Let $\mathbf{E} \subseteq \mathbf{V}$ denotes the set of evidence and $\mathbf{e}$ their values. The input to the algorithm is the BN and the evidence. The output are the conditional probabilities for the values of each node in $\mathbf{V\text{-}E}$ given $\mathbf{E} = e$. Finally, denote by $X_j \in \mathbf{V\text{-}E}$ and x_{jk} the *kth* value in the space of X_j's, $\mathbf{l}$ the number of nodes inside X_j space , $\mathbf{n}$ the number of nodes in $\mathbf{V}$ and $\mathbf{m}$ the number of trials.

Since the method samples values according to some probability, the conditional probability will not return the same exact result each time someone runs a query. As an example, considering a three discrete variables with a structure shown in fig. 2.2, a possible query is $P(KPI6 = 5K1|A16 = 0, A28 = 3)$ which is performed with a different number of trials on the logic sampling. The results are shown in fig. 2.3.

As the number of trials is increased, the resulting probabilities from the query, gather around a central value. The simulation was repeated two times, one with ten thousand trials and the second with one hundred thousand trials. In each run, the query was repeated five hundred times which is the sum of the counts in the histograms in fig. 2.3.

Algorithm 8: Pseudo code for logic sampling algorithm [29] .

1. **Ancestral ordering and Initialisation** : Order all the nodes in some ancestral order and initialise every x_{jk} to zero.

2. **for** *(i=1; i<= m; i++)* **do**
   { j=1;
   while *(j <= n)* **do**
      generate a value x'_j for X_j according to $P(x_j|pa_j)$(pa is the parent nodes of X_j).
      if *($X_j \in E$ and $x'_j \neq e_j$)* **then**
         j=1
      else
         j++
      end
   end
   for *(each X_j)* **do**
      for *(k=1: k<= l: k++)* **do**
         if *($x_{jk} == x'_j$)* **then**
            add 1 to occurrences of x_{jk}
         end
      end
   end
   }End first For
   end

3. **for** *(each X_j)* **do**
   for *(k=1: k<= l: k++)* **do**
      $\hat{P}(x_{jk}|e) = occurrences/m$
   end
   end

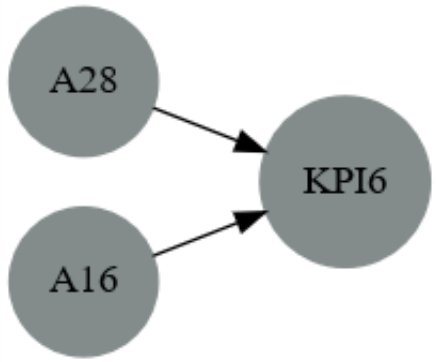


Figure 2.2: Example of three nodes representing the connections between three discrete variables.

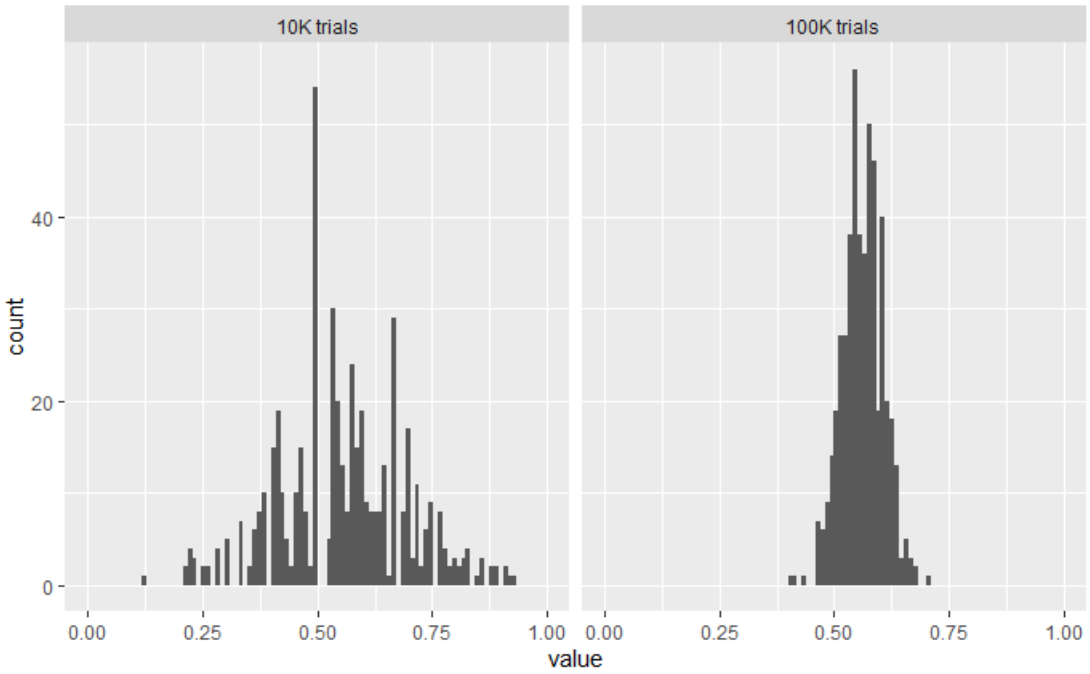


Figure 2.3: Query run using Logic sampling with 10,000 trials on the left and 100,000 on the right. Each run consisted of 500 repetitions of the query. The counts in the histogram, sum up to 500.

2.5 Bootstrap and Model Averaging

Since the estimation of a Bayesian Network Structure might be uncertain, some methods where proposed in order to give some degree of confidence. A simple application of a scoring algorithm to return the highest scored structure, might not be enough to produce some confidence on the existence e.g. of an arc between the variables $X \rightarrow Y$. In [30] the bootstrap approach was proposed as a technique to address this issue. In general, with bootstrap we treat our sample data as the "population" and by using sample with replacement, we create smaller bootstrap samples where we can make estimates like confidence intervals for the parameters

and such. There is a parametric and non-parametric version of bootstrap. In the parametric bootstrap, an assumption for the distribution of the data must be made. The bootstrap approach can by summarised as :

Algorithm 9: Bootsrtap method for arc confidence.

1. **for** $i = 1 \cdots n$ **do**
   (a) Create $\mathcal{D}_i^*$ bootstrap samples from the original dataset $\mathcal{D}$ using non parametric bootstrap (although parametric bootstrap can be used also).
   (b) Fop each $\mathcal{D}_i^*$ learn its structure $\mathcal{G}_i^* = (V, R_i)$.
   end
2. Compute the confidence of each possible arc α_i to be present in the true network $\mathcal{G}_0 = (V, A_0)$:

$$\hat{p}_i = \hat{P}(\alpha_i) = \frac{1}{B} \sum_{i=1}^{n} \mathbb{1}_{e_i \in A_i} \tag{2.17}$$

   which will add one each time a specific arc is present in the structures from the bootstrap datasets.

As explained in [31], they showed a simple rule in order to calculate an arc's strength and confidence of the direction from the bootstrap method. If we have B structures learned from the bootstrap samples created from our dataset D, then if t_1 is the number of occurrences of the arc $X- > Y$ and t_2 the corresponding number for the arc $Y- > X$ then the strength of an arc is given by $(t_1 + t_2/B)$. If $t_1 > t_2$ then it keeps the arc $X- > Y$ with confidence $t_1/(t_1 + t_2)$.

When the model averaging is performed, one can specify the significance threshold in a custom way. It is the threshold above which an arc is considered significant enough to be included in the network and it without taking into account for the direction. This parameter can pass into the model averaging phase in order to have the desired outcome.

$$A^* = \frac{1}{B} \sum_{i=1}^{B} \mathcal{A}_i^* \tag{2.18}$$

where A^* are the average adjacency matrix and $\mathcal{A}_i^*$ the adjacency matrices for each bootstrap estimated high scored structure. Then the matrix is filtered according to the threshold value :

$$A = (X_i \to X_j | A^* > \text{threshold}) \tag{2.19}$$

The non parametric bootstrap method is a frequentist's approach. An other approach worth mentioning, is the Bayesian one , the full Bayesian approach also, as described in [32] which uses Monte Carlo sampling from the posterior $P(\mathcal{G}|\mathcal{D})$.

Estimating the threshold in eq. (2.19) is usually based on ad-hoc choices but in [33] a statistical approach was proposed. The analysis is based on calculating the L_1 norm between the cumulative distribution function of the empirical strengths, as explained previously, and the ones calculated from the asymptotic case.

2.6 Summary

Structure learning is the first step for creating a DAG from data. In the years many algorithms were developed that fall in three categories: scored based approach, constraint and hybrid. The Hill Climbing search will perform a local search starting from an initial graph and it will find the structure that maximizes a scoring function. A great downside is that they can stack to local maxima. After the structure is learnt, then we perform the task of learning the parameters. One way to do this is to use the Maximum Likelihood method. In many cases an exact inference is intractable and the only choice is to try approximate methods. One very common algorithm for that is the Logic Sampling which is a Monte Carlo method. In order to create a more robust structure one can perform bootstrap sampling to create different structures and the do model averaging to produce a proposed structure.

3

Computational Results

First, some insights for the data and the domain are presented. Later, two different approaches are investigated of creating a Bayesian network from data. The data are naturally a mixed set of discrete and continuous data, but most of the discrete data had only one values which didn't contribute anything. Those kind of variables were removed from the variables pool. which comes with each one assumptions and restrictions. Later we are going to create a *discrete Bayesian network* by testing some techniques that will transform the continuous data into discrete ones. This approach also comes with serious drawbacks and mainly the loss of information that comes after discretising our variables.

3.1 Domain Insights

The variables used in the analysis belong mainly into to different categories : the variables that belong to class A that are more of fixed settings and the variables that belong to the KPIs class. The time resolution of the two different classes is different for in class A values are recorded once per day whereas the KPIs every fifteen minutes. Therefore in order to have a meaningful join, variables of class A keep the same value between observations. Further the KPIs are considered as a time series.

The variables of both classes belong to a hierarchy of structures. The most macroscopic structure considered here is a network of antennas grouped by region. Each antenna consists of different nested levels and the data describes the variables in each different level. Let $N = 1 \dots n$ the number of antennas and $L = 1 \dots m$ all the levels from all antennas, then we can see our variables as $X_{L:N}$ where $X = (A \cup KPIs)$ and it gives the value for variable **X** for level **L** in antenna **N**.

Both data sets contained missing values. Although the percentage was very small, they had to be taken care of in order to be able to run the algorithms. The rows containing missing values were deleted because the percentage was really small compared to the size of the dataset and would seem to have any negative affects.

Finally, data size was restricted mainly for two reasons : 1) In order to be able to perform computations locally and 2) because the measurements of the different data sets, didn't agreed in most of their time spans. A random re-sampling from the original data set reduced any auto-correlations effects due to the temporal nature

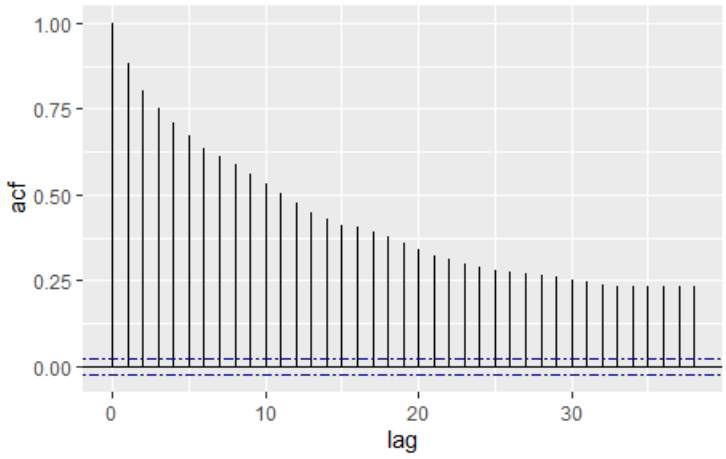


Figure 3.1: Auto correlation plot for one of the KPIs before re sampling. Since the vertical lines are way above the limits set by the horizontal lines, suggests strong correlation between the consecutive data points.

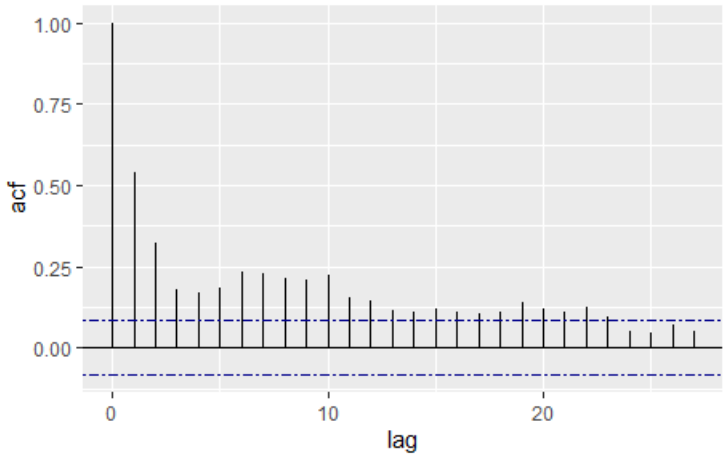


Figure 3.2: Auto correlation plot for one of the KPIs after re sampling. The auto correlation reduces over time due to random sampling of the data set.

of the KPIs.

3.2 Hybrid/Gaussian Model Results

The model variables($\mathbf{V}$) are separated to the sets of discrete(Δ) and continuous(Γ) such that $\mathbf{V} = \Delta \cup \Gamma$. If $\mathbf{Y}$ denotes the continuous variables and $\mathbf{I}$ the discrete ones, then the joint distribution of the continuous variables given the discrete ones, is assumed to be a multivariate Gaussian.

For calculating the structure of the Bayesian network, a variety of algorithms were tested but in the end HC algorithm was used. The choice to focus on the HC algorithm was done mainly for two reasons : 1) it was faster to study the output 2) can scale better according to literature. In fig. 3.3 we see the results. In order to

get better estimate for the structure, prior expert knowledge was utilised in order to exclude/include edges. Since HC is a score algorithm the Bayesian information criterion(BIC) was used to score structures in the the DAG space. Many nodes are left without connections and this is something domain experts expected, since they are not strongly related to KPIs.

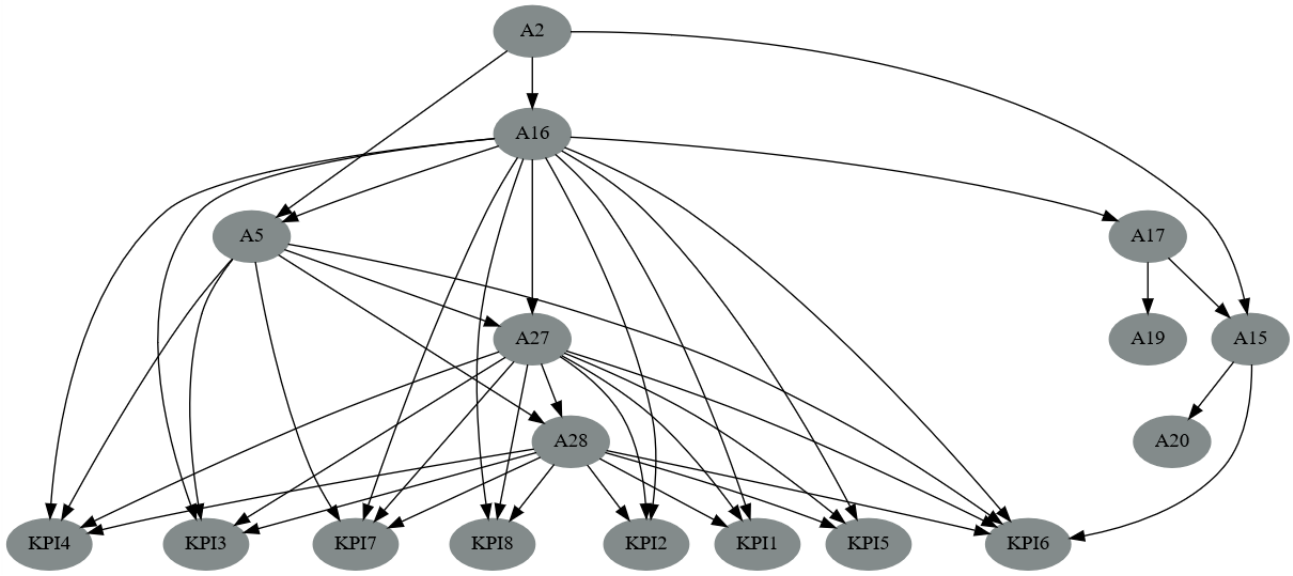


Figure 3.3: Structure estimation from mixed data using HC algorithm starting from an empty initial graph. The model is a CLGN where arcs from KPIs to A variables and arcs between the KPIs, where blacklisted due to domain knowledge.

Since the Hill Climbing search can stack in local maxima, we could improve the search by making random restarts with random initial permutations of the initial graph. Initially, the Hill Climbing search will start with random graph(a graph with no edges at all or a generated graph based on the variables, but with random arcs connecting them.) and it will terminate when the highest scoring DAG is found. When doing a restart, the initial graph will be altered by doing random permutations on the edges and the search will start again from a different initial point in the space of DAGs. As noted in previous chapters, since the algorithm does a local search, we cannot cover all the space of DAGs, but with the random restarts, more graphs will be examined. In fig. 3.4 we can see the results of running the Hill Climbing search together with 20 random restarts. In each restart 10 random permutations on the edges of the initial graph are performed.

Using the bootstrap approach now gives fig. 3.5 with an estimated threshold of about 0.5.
In order to get a robust estimation of the structure, the bootstrap implementation was let to run with 200 replications and sample sizes of the size of the dataset.

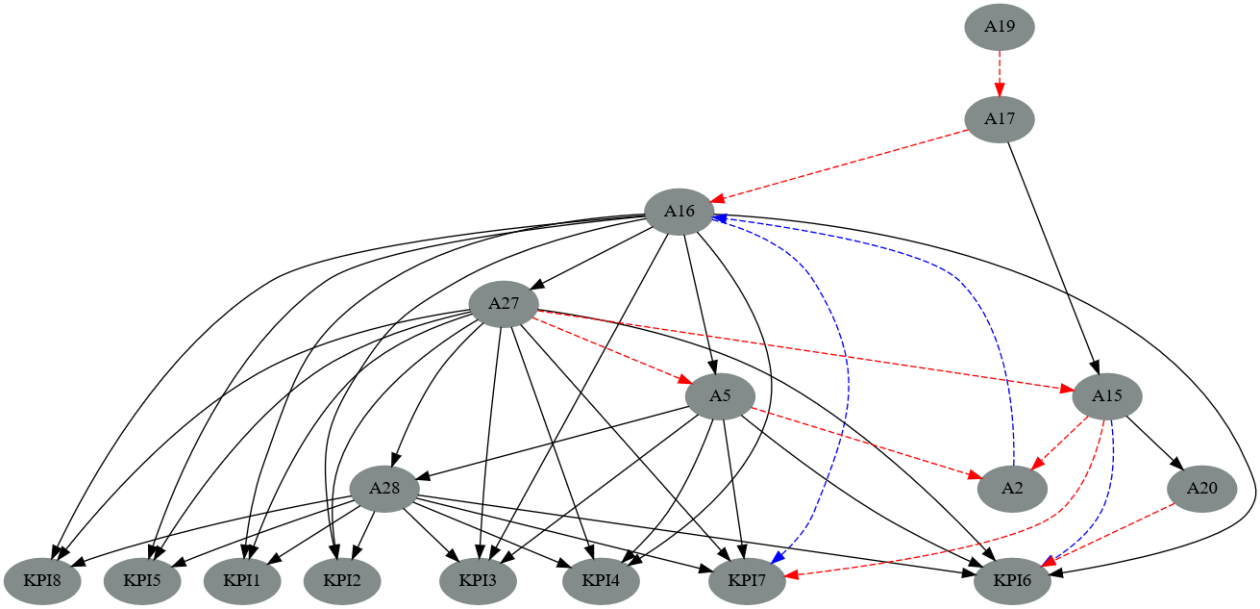


Figure 3.4: Structure estimation from mixed data using HC algorithm together with 10 random restarts, 20 perturbations on starting graphs on each restart and random initial graph. The red dashed lines are the false positive arcs and the blue dashed lines the false negatives.

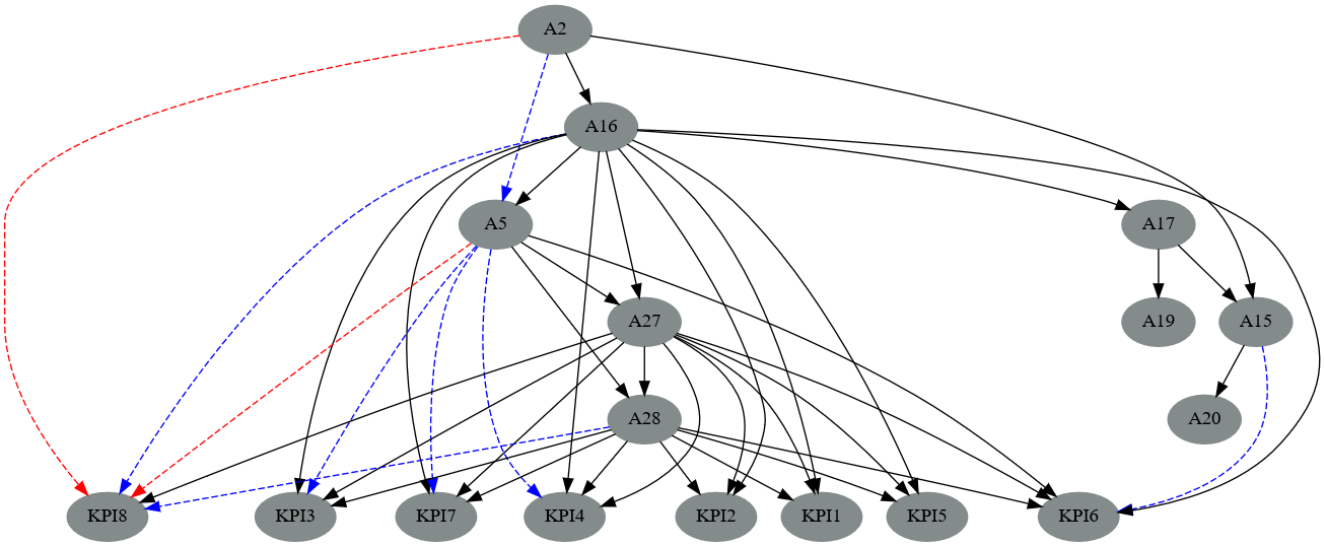


Figure 3.5: Structure estimation from mixed data using HC with bootstrap with 200 replications and a pre-estimated threshold. The red dashed lines are the false positive arcs whereas the blue dashed lines the false negative arcs.

3.3 Discrete Model

Most of the variables are either integers of certain value, categorical and continuous real numbers. Since lot of the integer variables were representing categories, they were directly treated as factor variables in R. The variables with small number of

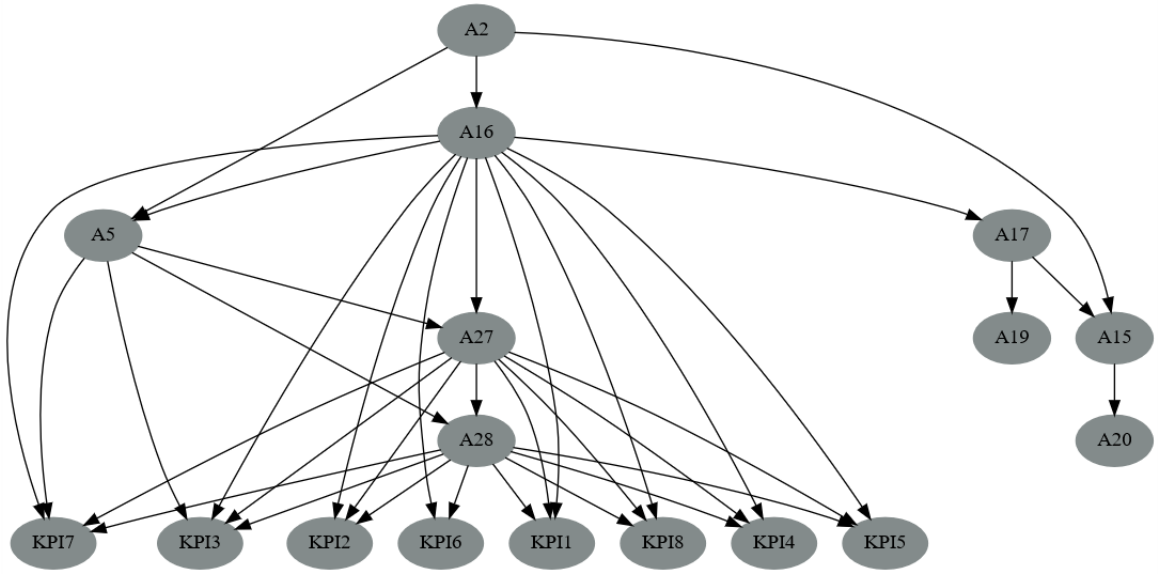


Figure 3.6: Structure learnt from data after discretization. The HC algorithm was used with random initial graph. The edges from KPIs to A variables were blacklisted together arcs between KPIs based on domain knowledge.

integer values, where treated as factors with the appropriate number of levels. Those where mostly the cases for the variables of class A and one of the KPIs. The rest of the KPIs where discretized using the hartemink method . Again, arcs going from KPIs to other variables of the class A were blacklisted together with arcs between KPIs.

3.3.1 Results

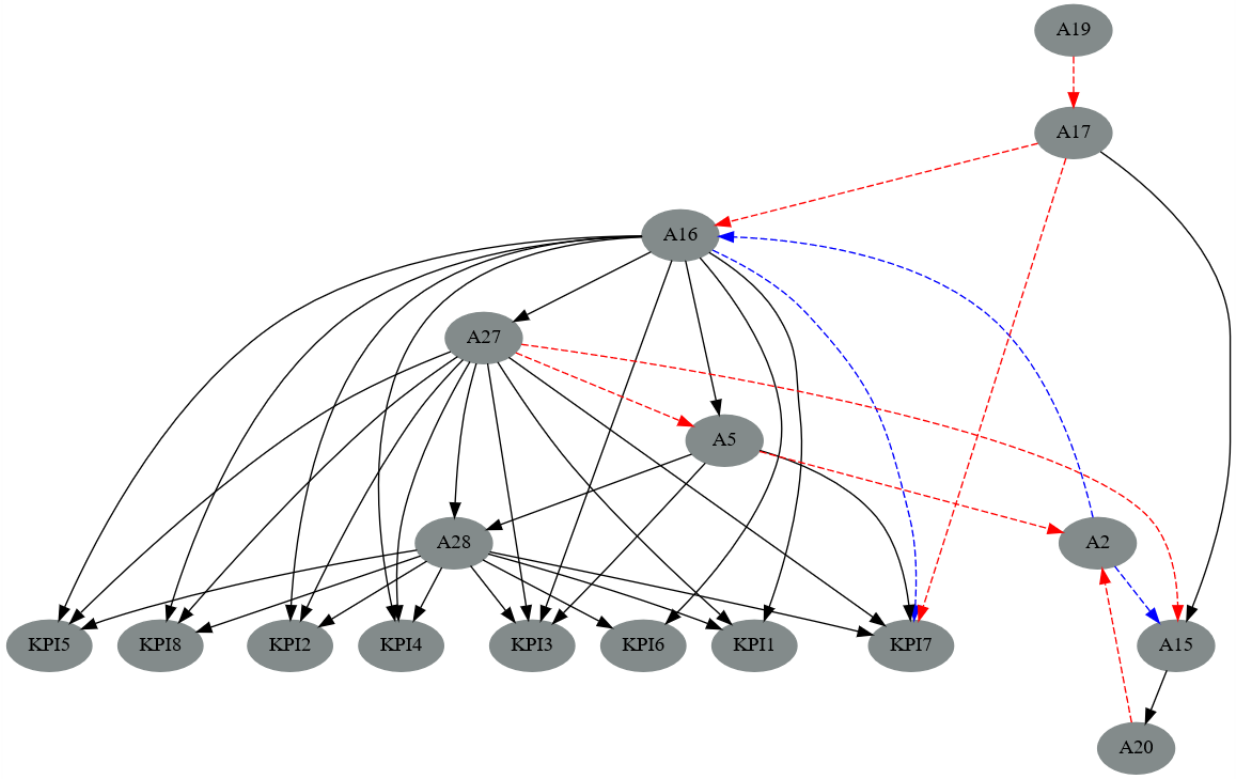


Figure 3.7: Structure learnt from discrete dataset. The HC algorithm used with 10 random restarts and 20 random permuted arcs after each restart. The red dashed lines correspond to the false positives arcs and with dashed blue lines the false negatives arcs.

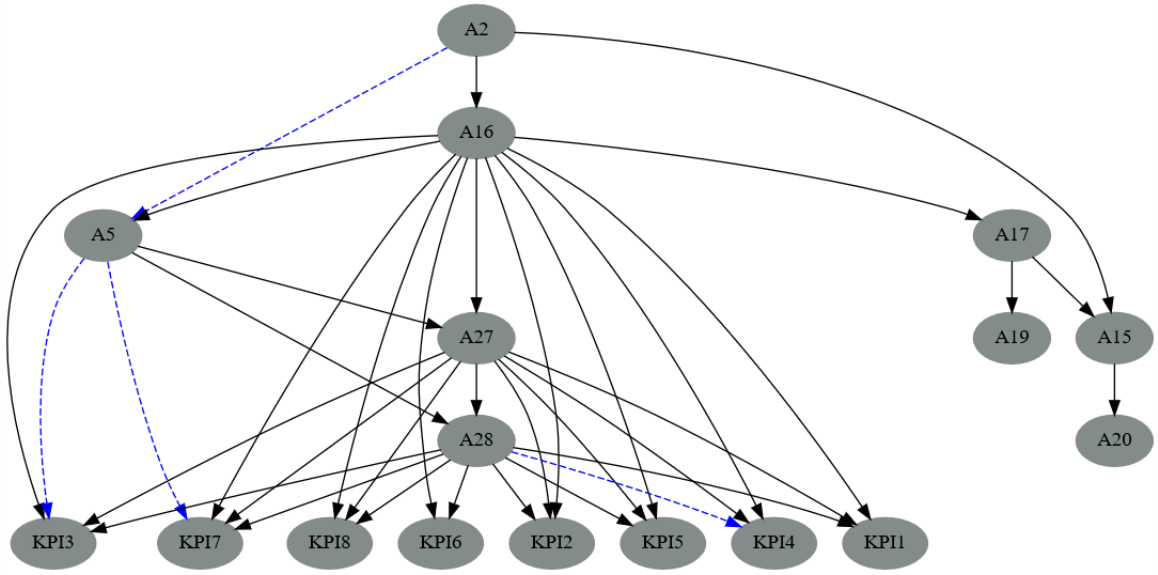


Figure 3.8: Structure learnt from discrete dataset. The HC algorithm used with bootstrap approach with 200 number of replicates.The blue dashed lines correspond to the false negatives arcs.

3.4 Discussion

Since one of the main goals of this thesis is the application of Bayesian networks in order to find possible causal relationships between the variables, an a priori structure is not known but more of insights depended on experts knowledge. There are significant differences between the two different approaches and there are valid restrictions that should be taken into account. The two strictures learnt with the plain HC with mixed data and discrete data don't differ a lot and the reason is that the black list of arcs permitted in the networks, coincides with the restriction that CLGN have from construction that discrete nodes cannot have continuous nodes as parents. Maybe the most important aspect is to what data you want to apply the model to. For this thesis, a small amount of the available variables was used and the effort was mostly to create a credible test case. The discrete model, although heavily dependent on the number and kind of bins, will allow more complex interactions between the variables although there will be information loss due to the binning. But still, it doesn't requires any normality assumption. On the other hand, much care should be put in order to transform a continuous variable into a discrete one and it requires a lot of knowledge on the data.

The plain implementation of HC algorithm, took almost one minute to finish on a mixed dataset of about half a million entries. Because of the different configurations of parents between the simple implementation and with the random restarts networks, the number of parameters learned was 684 and 732 respectively although they had the same number of edges. The bootstrap results converged to a very similar structure but with a bit smaller number of arcs. All of the models represent a conditional linear Gaussian Bayesian network. The number of parameters per node, is the number of its parents plus an intercept (see eq. (1.14)) for each configuration of the discrete parents plus any weights for cintinuous parents (see table E.1 also). In order to test the predictive capabilities of the models, a 10-fold cross validation was performed with the negative logarithm of the likelihood as the loss function. The were not significant differences between the three approaches with all the models reporting a value close(and around) to $28 - 31$.

In order to construct the discrete model, the variables of class A were transformed into factor by almost direct transformation of their values. In order to convert the KPIs to discrete variables, the hatermink method was used. The structures learnt with the discrete data had small differences from the structures learnt in the continuous case. One difference is that the discrete data produced a network with less adges(37 Vs 41). This is due to the fact that there is loss of information because of the discretization of the variables. Most of the computational time of learning the networks went to the structure learning. Calculating the structures for the discrete models, was much faster than the continuous ones. On possible explanation for this is that the CLGNs need to perform linear regressions which could have contributed to the overall running time. The implementation of the simple HC algorithm took about two seconds for the discrete case compared to almost one minute for the continuous model. The bootstrap approach, was implemented with various number of

replicate numbers but the structures were the same so a small number of 200 replicates was kept every time was needed to run the method.

The whole dataset is more complex than the test case used in this thesis. It seems more possible to try some discretization technique rather than applying a CLGN. This is mostly for the reason of the normality assumption and because continuous nodes cannot be parents to discrete nodes. Further an improved implementation can take into account the structure and 'hierarchy' of the data for a more accurate learning procedure.

Finally, since the data include time series, it is worth investigating the case of Dynamic Bayesian networks which are Bayesian networks for modelling time series data.

A

Data Plots

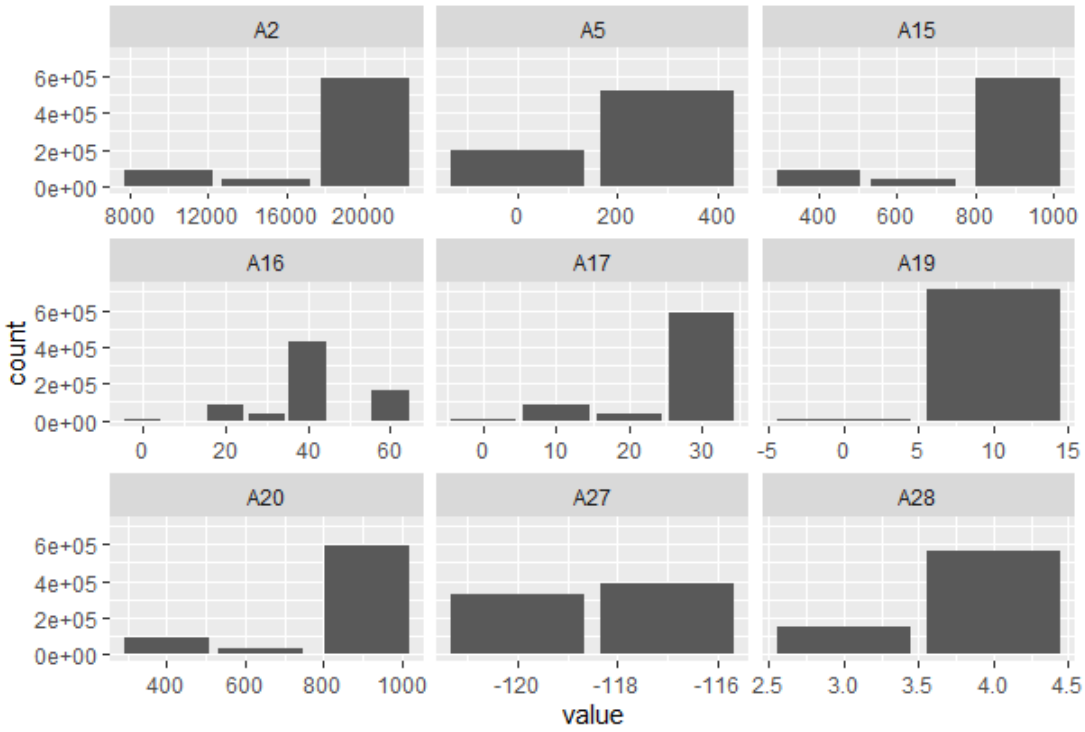


Figure A.1: Barplot of A variables.

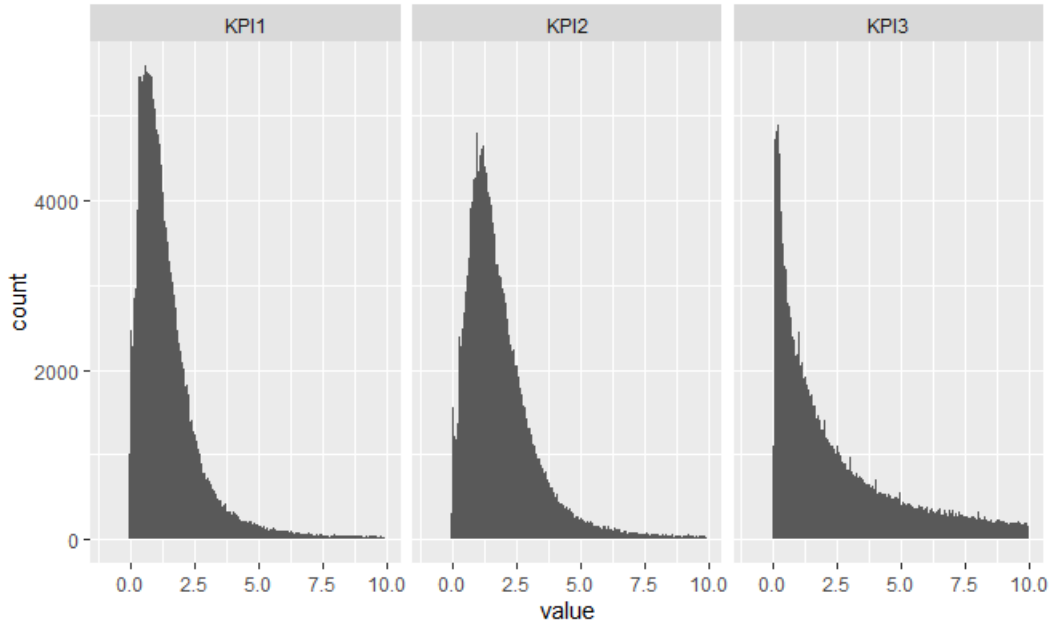


Figure A.2: Histogram of continuous variables before discretisation.

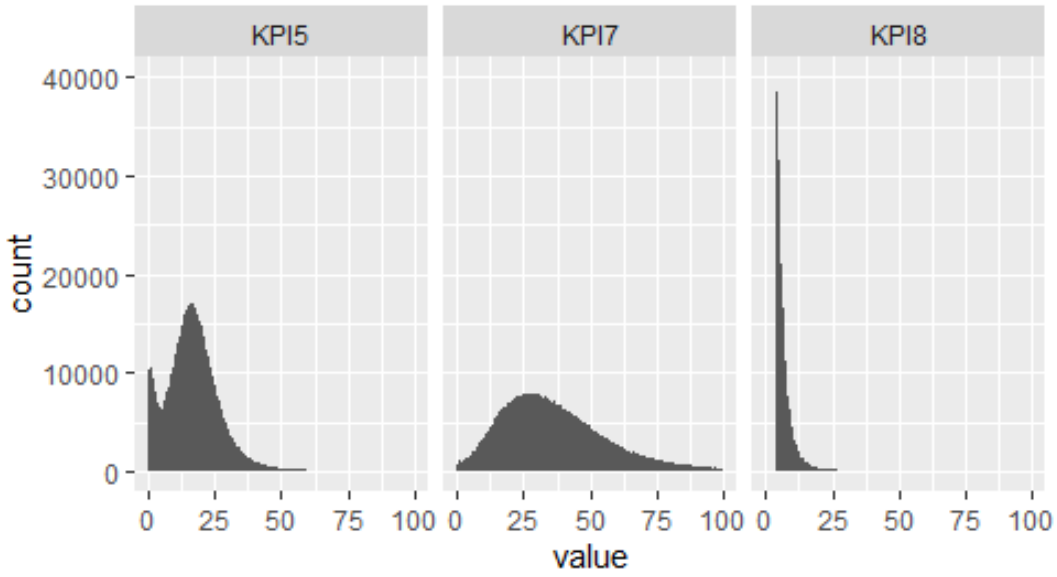


Figure A.3: Histogram of continuous variables before discretisation.

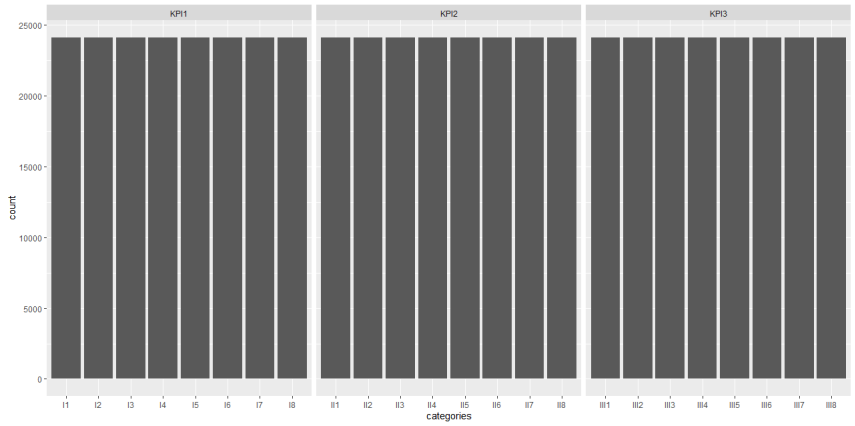


Figure A.4: Discretised variables with quantile method. Ir creates bins of equal frequency of points. Does not preserve initial distribution of data.

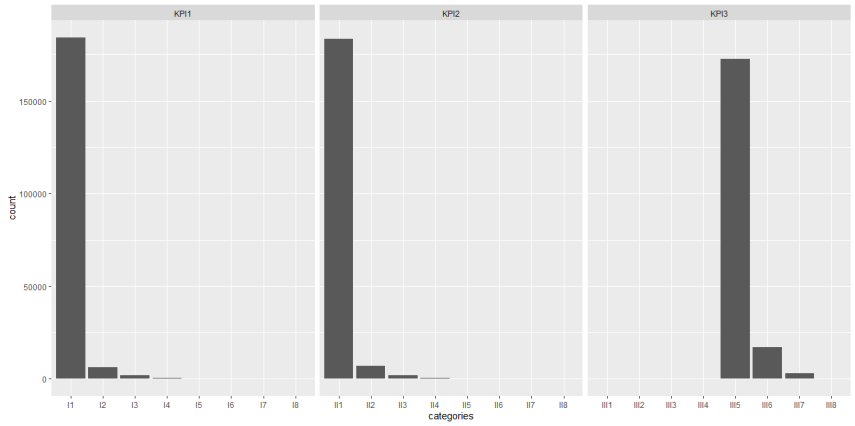


Figure A.5: Discretised variables with interval method. It creates uneven distribution of points in bins and it resembles the initial distribution.

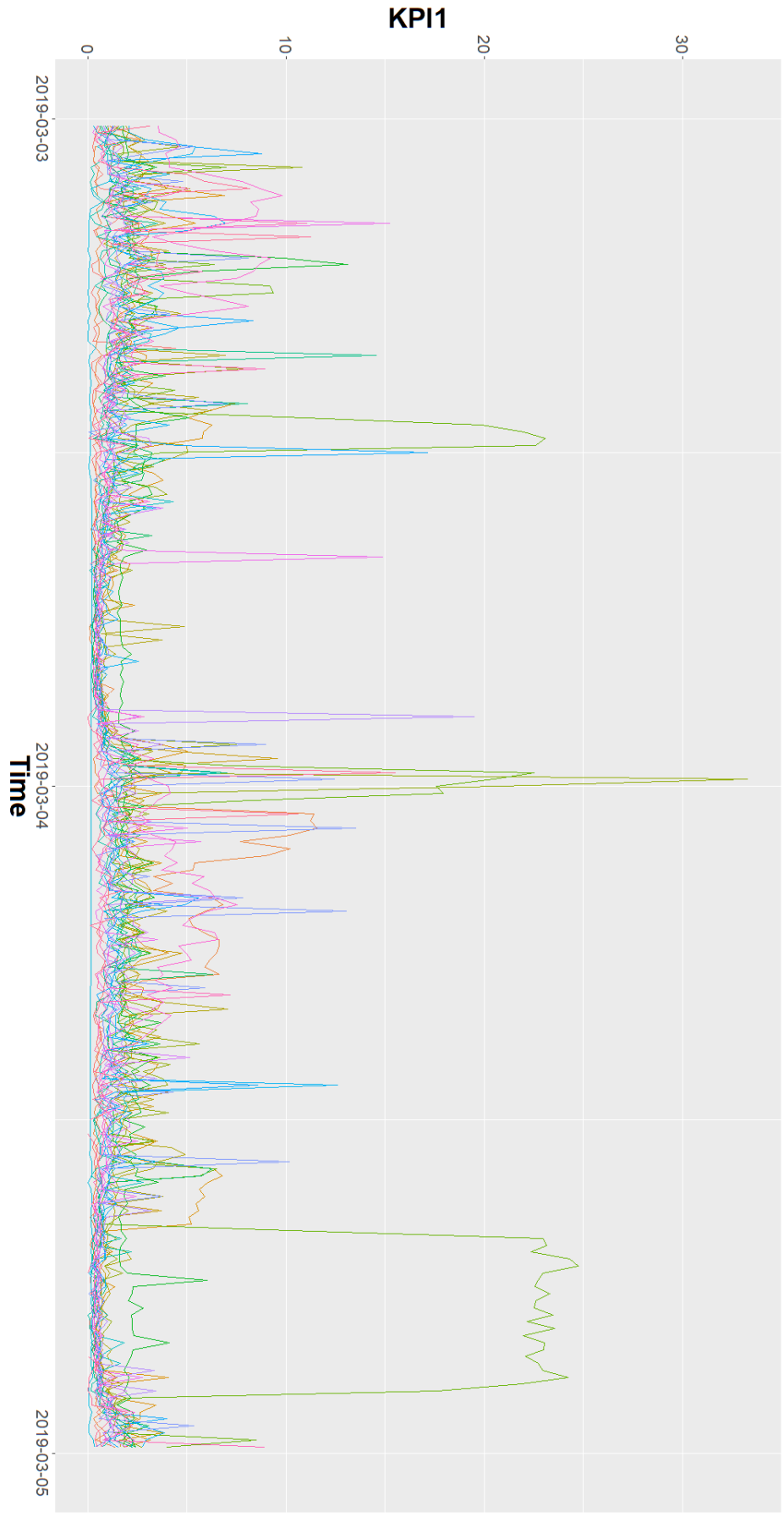


Figure A.6: Time evolution of the KPI1. Different colours correspond to different levels.

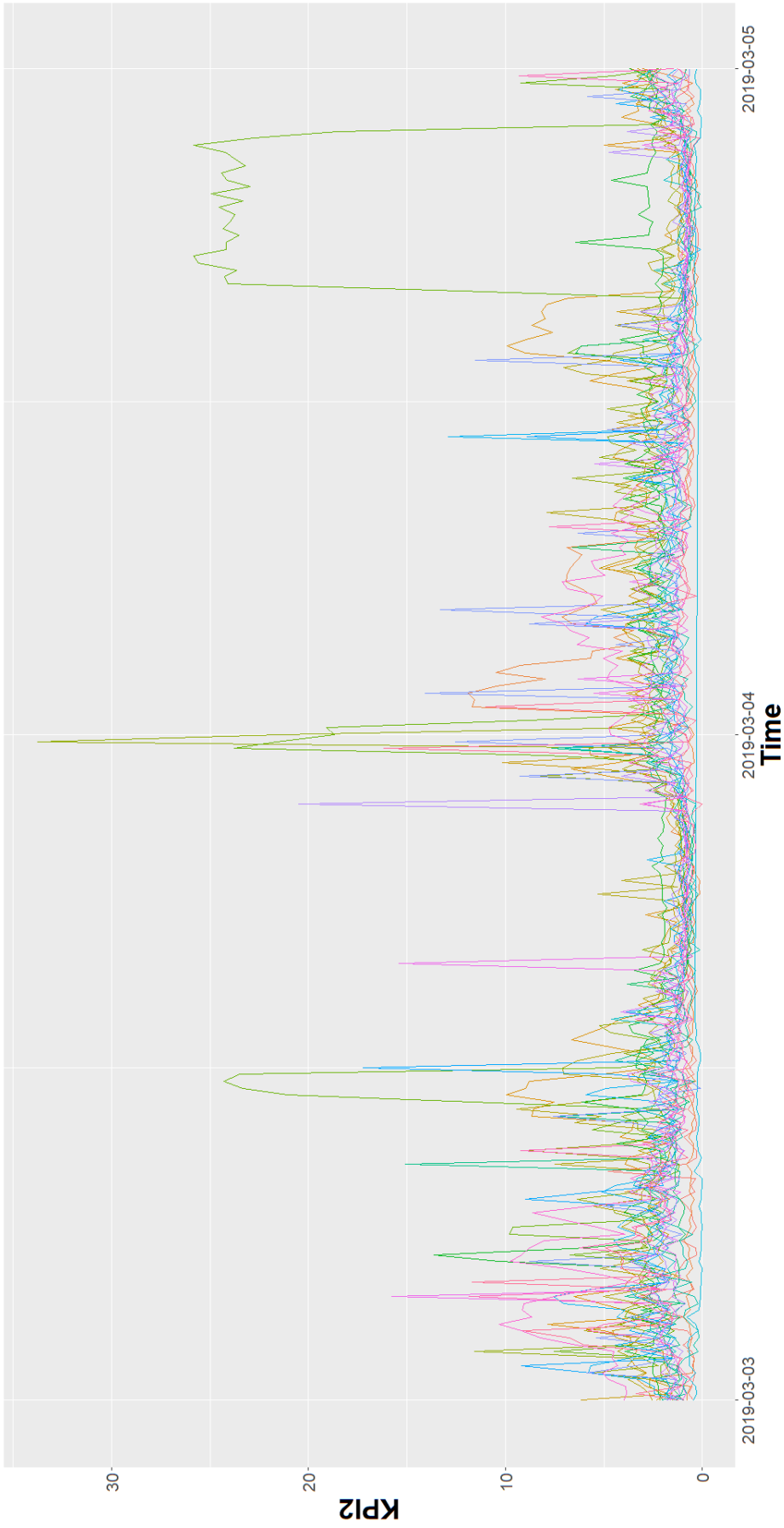


Figure A.7: Time evolution of the KPI2. Different colours correspond to different levels.

B

Discrete Data Tables

	Levels	Freq
A2.1	10000	66415
A2.2	15000	26474
A2.3	20000	407111
A5.1	0	133946
A5.2	300	366054
A15.1	400	66982
A15.2	640	26474
A15.3	910	406544
A16.1	0	3814
A16.2	20	65300
A16.3	30	26429
A16.4	40	293517
A16.5	60	110923
A17.1	0	3814
A17.2	10	65867
A17.3	20	26429
A17.4	30	403873
A19.1	0	3814
A19.2	10	496169
A20.1	400	66982
A20.2	640	26474
A20.3	910	406544
A27.1	-120	225378
A27.2	-117	274622
A28.1	3	99458
A28.2	4	400542

C

Hartemink Method

Here it follows an example to illustrate how the Hartemink discretization method works. Given an initial dataframe as in table C.1 :

	X	SY
1	5.10	3.50
2	4.90	3.00
3	4.70	3.20
4	4.60	3.10
5	5.00	3.60
6	5.40	3.90
7	4.60	3.40
8	5.00	3.40
9	4.40	2.90
10	4.90	3.10

Table C.1: Toy data frame based on the iris data set.

	X	Y
1	(4.9,5.15]	(3.4,3.65]
2	(4.65,4.9]	[2.9,3.15]
3	(4.65,4.9]	(3.15,3.4]
4	[4.4,4.65]	[2.9,3.15]
5	(4.9,5.15]	(3.4,3.65]
6	(5.15,5.4]	(3.65,3.9]
7	[4.4,4.65]	(3.15,3.4]
8	(4.9,5.15]	(3.15,3.4]
9	[4.4,4.65]	[2.9,3.15]
10	(4.65,4.9]	[2.9,3.15]

Table C.2: The dataframe after the initial discretization using the interval method. The variables are binned into four levels each.

then, the dataframe shown in table C.2 is used as input to the Hartemink method. The method starts with an initial discretization number of levels (here it is four) and ends when the desired final number of discretization levels is reached (e.g. two in this case) which is smaller than the initial number. Then a loop visits one variable at time. Starting with variable X : it has the four levels, in order,

$[4.4, 4.65], (4.65, 4.9], (4.9, 5.15], (5.15, 5.4]$. The method will start to collapse the adjacent levels and compute the MI in each case :

1. Creating three levels.
   (a) Collapse $[4.4, 4.65], (4.65, 4.9]$ to become one level $[4.4, 4.65]$. the new dataframe is then table C.3 :

	X	Y
1	(4.9,5.15]	(3.4,3.65]
2	[4.4,4.65]	[2.9,3.15]
3	[4.4,4.65]	(3.15,3.4]
4	[4.4,4.65]	[2.9,3.15]
5	(4.9,5.15]	(3.4,3.65]
6	(5.15,5.4]	(3.65,3.9]
7	[4.4,4.65]	(3.15,3.4]
8	(4.9,5.15]	(3.15,3.4]
9	[4.4,4.65]	[2.9,3.15]
10	[4.4,4.65]	[2.9,3.15]

Table C.3: The dataframe after the after collapsing $[4.4, 4.65], (4.65, 4.9]$.

   (b) Compute $MI(X, Y) = 0.7069$ based on eq. (1.8)
   (c) Collapse $(4.65, 4.9], (4.9, 5.15]$, the new dataframe is now table C.4 :

	X	Y
1	(4.65,4.9]	(3.4,3.65]
2	(4.65,4.9]	[2.9,3.15]
3	(4.65,4.9]	(3.15,3.4]
4	[4.4,4.65]	[2.9,3.15]
5	(4.65,4.9]	(3.4,3.65]
6	(5.15,5.4]	(3.65,3.9]
7	[4.4,4.65]	(3.15,3.4]
8	(4.65,4.9]	(3.15,3.4]
9	[4.4,4.65]	[2.9,3.15]
10	(4.65,4.9]	[2.9,3.15]

Table C.4: The dataframe after the after collapsing $(4.65, 4.9], (4.9, 5.15]$.

   (d) Compute $MI(X, Y) = 0.4297$
   (e) Collapse $(4.9, 5.15], (5.15, 5.4]$, the new dataframe is now table C.5 :
   (f) Compute $MI(X, Y) = 0.48205$
   (g) Keep the collapsing with the highest MI score (which is the first one in this example). Use table C.3 is input to the next phase. Variable has now three levels and the goal is to reach two levels
2. Creating two levels. Repeat the above procedure but using now table C.3. Keep the collapse that scores the highest mutual information. Since the target number of discretization levels in this example is two, it finishes the calculations for the variable X in this step.

	X	Y
1	(4.9,5.15]	(3.4,3.65]
2	(4.65,4.9]	[2.9,3.15]
3	(4.65,4.9]	(3.15,3.4]
4	[4.4,4.65]	[2.9,3.15]
5	(4.9,5.15]	(3.4,3.65]
6	(4.9,5.15]	(3.65,3.9]
7	[4.4,4.65]	(3.15,3.4]
8	(4.9,5.15]	(3.15,3.4]
9	[4.4,4.65]	[2.9,3.15]
10	(4.65,4.9]	[2.9,3.15]

Table C.5: The dataframe after the after collapsing $(4.9, 5.15], (5.15, 5.4]$.

3. When the first variable reached the desired number of discretization levels, the algorithm is repeated for the next variable in the dataset and performing the exact same steps as before. It starts with the final dataset produced in the previous steps.

In this example there were two variables. If we had more variables, e.g. X, Y, Z, W, each time an MI needs to be calculated, it is the sum of MI(X,Y), MI(X,Z), M(X,W) that is used for comparing which collapse is better.

D

Maximum likelihood

Given the data set :

	A	B	C
1	b	c	b
2	b	a	c
3	a	a	a
4	a	a	a
5	a	a	b
6	c	c	a
7	c	c	b
8	b	b	a
9	b	b	b
10	b	a	b

a possible DAG is $A->B$. The conditional probability table for node B would be then :

	$A=a$	$A=b$	$A=c$
$B=a$	1	0	0
$B=b$	0	0.4	0
$B=c$	0	0.2	1

since, for example $\theta_{B=b|A=b}=0.4$ is given by :

$$\frac{\#\text{of occurrences were} B=b|A=b}{\#\text{of occurrences were} A=b}=\frac{2}{5} \tag{D.1}$$

or $\theta_{B=c|A=b}=0.2$ is given by :

$$\frac{\#\text{of occurrences were} B=c|A=b}{\#\text{of occurrences were} A=b}=\frac{1}{5} \tag{D.2}$$

E

Number of parameters

This chapter gives an example of how to calculate the number parameters on a conditional linear Gaussian Bayesian network.

Given the toy dataset, as shown below, of mixed type of variables

	A	B	C	D	E	F	G	H
1	a	b	d	6.46	11.99	b	34.84	2.33
2	b	a	a	12.76	30.44	b	106.64	2.36
3	b	c	c	12.18	17.22	a	68.93	2.32
4	b	c	d	12.01	14.42	b	86.18	2.42
5	b	a	a	12.33	30.40	b	103.59	2.27
6	b	c	c	12.61	15.19	b	90.85	2.31
7	b	a	a	12.09	28.83	b	100.75	2.07
8	b	b	c	12.40	18.37	a	71.11	2.41
9	b	c	b	12.01	17.01	a	68.20	2.32
10	b	c	a	12.33	15.50	a	68.10	2.29

we learn the DAG :
The only discrete variables are A, B, C, F with number of levels (or values) 2, 3, 4, 2 respectively. The node of variable A doesn't have any parents and since it is

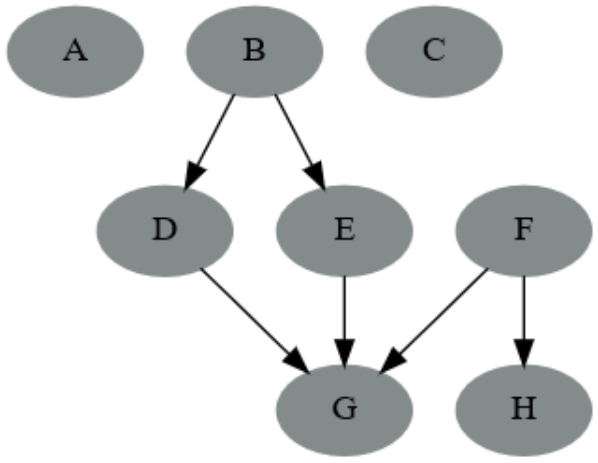


Figure E.1: Toy dataset with mixed type of variables

a discrete variable with 2 levels, the number of parameters is $(2-1)2^0 = 1$[1]. The node of the continuous variable D has one discrete parent only (variable B). Thus according to eq. (1.13) it needs to learn only the intercept and the residual error for each value of the discrete parent. This means $2 * 3 = 6$ parameters. The node for the continuous variable G has three parents: the continuous variables D, E and the discrete variable F. So it is modelled like:

$$G_{F=a} = \beta_{0:a} + D\beta_D + E\beta_E + \epsilon_a \tag{E.1}$$

$$G_{F=b} = \beta_{0:b} + D\beta_D + E\beta_E + \epsilon_b \tag{E.2}$$

thus, there are four parameters in each equation and there are two equations for each value of the discrete parent. This gives a total of eight parameters. The number of parameters for the whole network are 33.

Variable	A	B	C	D	E	F	G	H
Number of parameters	1	2	3	6	6	1	8	6

Table E.1: Number of parameters learnt for each variable in the toy DAG.

[1]for a discrete BN, the number of free parameters is $(L-1)L^p$, p: number of parents and L: number of values

References

[1] Kevin P. Murphy. *Machine learning : a probabilistic perspective*. Cambridge, Mass. [u.a.]: MIT Press, 2013. ISBN: 9780262018029 0262018020. URL: https://www.amazon.com/Machine-Learning-Probabilistic-Perspective-Computation/dp/0262018020/ref=sr_1_2?ie=UTF8&qid=1336857747&sr=8-2.

[2] Susan Vineberg. "Dutch Book Arguments". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Spring 2016. Metaphysics Research Lab, Stanford University, 2016.

[3] Judea Pearl and Stuart Russell. "Bayesian Networks". In: (2002). URL: https://people.eecs.berkeley.edu/~russell/papers/hbtnn-bn.pdf.

[4] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. 2016. URL: http://arxiv.org/abs/1511.06434.

[5] Andrej Karpathy. *The Unreasonable Effectiveness of Recurrent Neural Networks*. 2015. URL: http://karpathy.github.io/2015/05/21/rnn-effectiveness/ (visited on 05/31/2015).

[6] Nir Friedman. "Inferring Cellular Networks Using Probabilistic Graphical Models". In: *Science* 303.5659 (2004), pp. 799–805. ISSN: 0036-8075. DOI: 10.1126/science.1094068. eprint: https://science.sciencemag.org/content/303/5659/799.full.pdf. URL: https://science.sciencemag.org/content/303/5659/799.

[7] Dominik Aronsky and Peter J. Haug. "Diagnosing community-acquired pneumonia with a Bayesian network". In: *Proceedings. AMIA Symposium* (1998), pp. 632–6.

[8] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988. ISBN: 1558604790.

[9] Ross D. Shachter. "Bayes-Ball: The Rational Pastime (for Determining Irrelevance and Requisite Information in Belief Networks and Influence Diagrams)". In: *CoRR* abs/1301.7412 (2013). arXiv: 1301.7412. URL: http://arxiv.org/abs/1301.7412.

[10] Thomas Verma and Judea Pearl. "Equivalence and Synthesis of Causal Models". In: *Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence*. UAI '90. New York, NY, USA: Elsevier Science Inc., 1991, pp. 255–270. ISBN: 0-444-89264-8. URL: http://dl.acm.org/citation.cfm?id=647233.719736.

[11] Eunice Yuh-Jie Chen, Arthur Choi, and Adnan Darwiche. "Enumerating Equivalence Classes of Bayesian Networks using EC Graphs". In: *AISTATS*. 2016.

[12] Dana Pe'er. "Bayesian Network Analysis of Signaling Networks: A Primer". In: *Science's STKE : signal transduction knowledge environment* 2005 (May 2005), pl4. DOI: 10.1126/stke.2812005pl4.

[13] Alexander John Hartemink. "Principled computational methods for the validation discovery of genetic regulatory networks". PhD thesis. Massachusetts Institute of Technology, 2001.

[14] Marco Scutari and Jean-Baptiste Denis. *Bayesian Networks with Examples in R*. ISBN 978-1-4822-2558-7, 978-1-4822-2560-0. Boca Raton: Chapman and Hall, 2014.

[15] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009. ISBN: 0262013193, 9780262013192.

[16] David Maxwell Chickering. "Learning Bayesian Networks is NP-Complete". In: *AISTATS*. 1995.

[17] Jin Tian and Ru He. "Computing Posterior Probabilities of Structural Features in Bayesian Networks". In: *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. UAI '09. Montreal, Quebec, Canada: AUAI Press, 2009, pp. 538–547. ISBN: 978-0-9749039-5-8. URL: http://dl.acm.org/citation.cfm?id=1795114.1795177.

[18] Markus Kalisch et al. "More Causal Inference with Graphical Models in R Package pcalg". In: 2014.

[19] Marco Scutari. "Measures of Variability for Graphical Models". In: 2011.

[20] Peter Spirtes and Clark Glymour. "An Algorithm for Fast Recovery of Sparse Causal Graphs". In: 1991.

[21] Dimitris Margaritis. *Learning Bayesian Network Model Structure From Data*. Tech. rep. 2003.

[22] Ioannis Tsamardinos, Constantin F. Aliferis, and Alexander R. Statnikov. "Algorithms for Large Scale Markov Blanket Discovery". In: *FLAIRS Conference*. 2003.

[23] Sandeep Yaramakala and Dimitris Margaritis. "Speculative Markov blanket discovery for optimal feature selection". In: *Fifth IEEE International Conference on Data Mining (ICDM'05)* (2005), 4 pp.-.

[24] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 3rd. Upper Saddle River, NJ, USA: Prentice Hall Press, 2009. ISBN: 0136042597, 9780136042594.

[25] Marco Scutari, Claudia Vitolo, and Allan Tucker. "Learning Bayesian networks from big data with greedy search: computational complexity and efficient implementation". In: *Statistics and Computing* 29.5 (Sept. 2019), pp. 1095–1108. ISSN: 1573-1375. DOI: 10.1007/s11222-019-09857-1. URL: https://doi.org/10.1007/s11222-019-09857-1.

[26] Bryan Andrews, Joseph Ramsey, and Gregory F. Cooper. "Scoring Bayesian networks of mixed variables". In: *International Journal of Data Science and Analytics* 6.1 (Aug. 2018), pp. 3–18. ISSN: 2364-4168. DOI: 10.1007/s41060-017-0085-7. URL: https://doi.org/10.1007/s41060-017-0085-7.

[27] Ivar Simonsson and Petter Mostad. "Exact Inference on Conditional Linear Γ-Gaussian Bayesian Networks". In: *Proceedings of the Eighth International Conference on Probabilistic Graphical Models*. Ed. by Alessandro Antonucci, Giorgio Corani, and Cassio Polpo Campos. 2016, pp. 474–486.

[28] Max HENRION. "Propagating Uncertainty in Bayesian Networks by Probabilistic Logic Sampling". In: *Uncertainty in Artificial Intelligence*. Ed. by John F. LEMMER and Laveen N. KANAL. Vol. 5. Machine Intelligence and Pattern Recognition. North-Holland, 1988, pp. 149–163. DOI: https://doi.org/10.1016/B978-0-444-70396-5.50019-4. URL: http://www.sciencedirect.com/science/article/pii/B9780444703965500194.

[29] Richard E. Neapolitan. *Learning Bayesian Networks*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2003. ISBN: 0130125342.

[30] Nir Friedman, Moisés Goldszmidt, and Abraham J. Wyner. "Data Analysis with Bayesian Networks: A Bootstrap Approach". In: *CoRR* abs/1301.6695 (2013). arXiv: 1301.6695. URL: http://arxiv.org/abs/1301.6695.

[31] Takeshi Kamimura et al. "Bootstrap analysis of gene networks based on Bayesian networks and nonparametric regression". In: *Genome Informatics* 14 (Jan. 2003), pp. 350–351.

[32] Nir Friedman and Daphne Koller. "Being Bayesian About Network Structure. A Bayesian Approach to Structure Discovery in Bayesian Networks". In: *Machine Learning* 50.1 (Jan. 2003), pp. 95–125. ISSN: 1573-0565. DOI: 10.1023/A:1020249912095. URL: https://doi.org/10.1023/A:1020249912095.

[33] Marco Scutari and Radhakrishnan Nagarajan. "Identifying significant edges in graphical models of molecular networks". In: *Artificial Intelligence in Medicine*. 2011.