The influence of learning algorithms for Bayesian Networks on predictions - a citation analysis study case

Dutch title: De invloed van leeralgoritmes voor Bayesiaanse netwerken op predicties- een citatie analyse casus

by

Redouan Ochalhi

to obtain the degree of Bachelor of Science in Applied Mathematics
at the Delft University of Technology,
to be defended publicly on Thursday July 11, 2019 at 02:00 PM.

Student number:	4582764	
Project duration:	April 1, 2019 – July 1, 2019	
Thesis committee:	Dr. ir. T. Nane,	TU Delft, supervisor
	Dr. B. van den Dries	TU Delft
	Dr. ir. R. van der Toorn	TU Delft

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Acknowledgements

I would like to thank my supervisor Tina Nane for the support during the whole process.

Abstract

In this thesis, attention is paid to building different Bayesian networks. You can think of aspects such as parameter learning, search procedures and score functions. In addition, a distinction is made between the use of Discrete Bayesian Networks and Gaussian Networks. These models both have different assumptions which are also discussed. Finally, the theory is applied to publication and citation data for a group of Canadian researchers. We will build Bayesian networks with different techniques and try to predict and compare the performance of researchers. We will also build an algorithm based on clustering that can perform predictions by using one of the possible learning algorithms.

Contents

Chapter 1

Outline of the thesis

A Bayesian network is used for two purposes: to do predictions and to understand the probabilistic relationship between different variables. It is a directed acyclic graph in which the nodes describe random variables , and the arcs the relations between them. To construct such a network, we could make use of learning algorithms like the Tabu algorithm and Hill-Climbing algorithm. These algorithms try to build a network that corresponds best to reality. In this research, We will mainly focus on comparing the predictive performance of the networks that are built by different algorithms. In the following paragraphs, you can read how the thesis is structured.

As said, in this thesis we will try to make predictions with the help of Bayesian Networks. Before we move on to this, it is good to have a solid basis. In Chapter 2 we therefore give an extensive introduction to Bayesian Networks which is necessary to understand the rest of this thesis. In chapter 2 we will also discuss topics such as learning algorithms and parameter learning.

In Chapter 3 and 4 we describe two different types of Bayesian Networks: Discrete Bayesian Networks and Gaussian Bayesian Networks. In Chapter 3 there are some discretization algorithms that are discussed and also the ways we can draw conclusions from Discrete Bayesian networks. In chapter 4 we will discuss what conditions we must meet to make use of Gaussian Bayesian networks and some tests to check whether we meet this conditions.

In the final chapter, we will apply the theory to a data set of Canadian researchers. We will try to construct different Bayesian Networks with different algorithms and compare them in some way. We will use the Mean Squared Error (MSE) for this purpose. In addition, we will also look at the bootstrap procedure and also apply this concept to our data set. Finally, we look at the predictive power of a self-produced algorithm based on discretization.

Chapter 2

Introduction in Bayesian Network theory

”Probabilistic graphical models are an elegant framework which combines uncertainty (probabilities) and logical structure (independence constraints) to compactly represent complex, real-world phenomena. The framework is quite general in that many of the commonly proposed statistical models (Bayesian networks, hidden Markov models, Ising models) can be described as graphical models. Graphical models have enjoyed a surge of interest in the last two decades, due both to the flexibility and power of the representation and to the increased ability to effectively learn and perform inference in large networks.” [1]

A Bayesian network is a graphical tool that is used to model probabilistic reasoning [2]. It is a directed acyclic graph in which the nodes describe random variables , and the arcs the relations between them. Usually the direction from an arc is seen as a causal relationship between the different events, but this is not necessarily the case. However, the causal interpretation is often a good intuitive way to read a probabilistic network. Bayesian networks make inference easier and ensure that we can understand the underlying interaction between different variables a lot better. In the remainder of this report it will become clear why this is the case.

To ensure that you can read this report without having much trouble understanding it, it is imperative that you have some affinity with probability theory and graph theory. I have listed a number of features for you below so that you can better understand the remainder of the report.

2.1 Basic definitions

Definition 2.1.0.1. *Suppose we have two events A and B. The conditional probability of A given B is defined by*

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)} \tag{2.1}$$

Definition 2.1.0.2. *Two events A and B are called conditionally independent given C if and only if $P(A \mid B, C) = P(A \mid C)$ and $P(B \mid A, C) = P(B \mid C)$.This is denoted as follows: $A \perp B \mid C$.*

There are a number of properties that we can easily deduce from the previous definitions. The following properties hold:

- **Symmetry:** For events A and B : $A \perp B \Rightarrow B \perp A$
- **Decomposition:** For events A,B and C : $A \perp B, C \Rightarrow A \perp B$ and $A \perp C$
- **Weak** For events A,B and C : $A \perp B, C \Rightarrow A \perp B \mid C$ and $A \perp C \mid B$
- **Contraction:** For events A,B and C : $A \perp B \mid C$ and $A \mid C \Rightarrow A \perp B, C$
- **Intersection** For events A,B,C and D : $A \perp B \mid C, D \Rightarrow A \perp D \mid C, B$

Definition 2.1.0.3 (DAG). *A directed graph that has no directed cycles is called a directed acyclic graph; this is often abbreviated to DAG.*

Definition 2.1.0.4. *A graph G is called **undirected**, if the edges of G have no direction.*

Definition 2.1.0.5. *A node A is called a **parent** of another node B if there is an edge in the network from A to B. In this case, B is called a **child** of A. The **descendants** of A are all the nodes that can be reached from A in the DAG.*

2.2 Bayesian networks

Bayesian networks are graphical representations of conditional independencies between different variables via DAGs. The nodes in such a graph represent random variables. Usually the direction of the edges in such a graph is seen as a causal relationship between the different events, but this is not necessarily true. We can read dependencies between different variables in such a graph by using the properties that will follow below.

Definition 2.2.0.1. *Formally, a **Bayesian network** consists of two parts. On the one hand a DAG and on the other a set of conditional probability distributions for each variable (node) conditioned on the parents of this variable (node).*

So there must be a link between the conditional independencies and the structure of a DAG. The following three definitions provide us the way to read such a network.

Definition 2.2.0.2. *The three possible connections are as follows:*
Serial connection: $X_i \rightarrow X_k \rightarrow X_h$
Divergent connection: $X_i \leftarrow X_k \rightarrow X_h$
Convergent connection: $X_i \rightarrow X_k \leftarrow X_h$

Definition 2.2.0.3. *If A,B and C are three disjoint subsets of the nodes of a DAG G, C* ***D-seperates*** *A from B* $(A \perp_G B \mid C)$*, if along every path between a node in A and a node in B in the undirected version of G, there exist a node v satisfying one of the following two conditions:*

(1) v has converging edges and v and its descendants are not in C
(2) is in C and does not have converging arcs

Definition 2.2.0.4. *A graph G is an* ***I-map*** *if* $(A \perp_G B \mid C) \Rightarrow (A \perp B \mid C)$

We are now ready to introduce the formal definition of a Bayesian Network.

Definition 2.2.0.5. *A Bayesian network is a minimal I-map.*

This means that when we remove any edge of the graph that belongs to the Bayesian network, the I-Map condition does not hold anymore. Since a Bayesian Networks is an I-map, this guarantees that we always deal with DAGs.

It is possible to decompose the global distribution of a multivariate random variable $X = (X_1, X_2, .., X_n)$ in the following way by using definition 2.1.0.1 :

$$P(\mathbf{X}) = \prod_{i=1}^{n} P(X_i | X_{i+1}, ...X_n) \tag{2.2}$$

The following definition can be easily deduced from the fact that an Bayesian Network is a minimal I-map.

Definition 2.2.0.6 (Local Markov property)**.** *Each node of a graph that belongs to a Bayesian network is conditionally independent of its non-descendants given its parents.*

By combining equation 2.2 and the Local Markov Property we are now able to decompose the global distribution of a multivariate random variable $\mathbf{X}$ in the following way:

$$P(\mathbf{X}) = \prod_{i=1} P(X_i | Parents(X_i)) \tag{2.3}$$

For example, the Local Markov property shows us that in Figure 2.1, node D is conditionally independent of all the other nodes given node C. We can derive the global distribution for this network according to (2.3) in the following way:

$$P(A, B, C, D, E) = P(A)P(B)P(D|C)P(E|B, C)P(C|A)$$

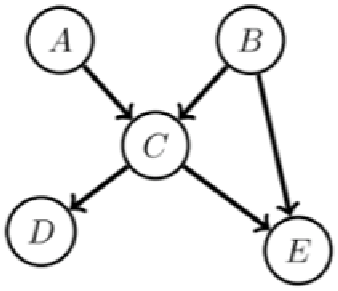


Figure 2.1: Bayesian Network of 5 variables

It can be shown that the global distributions for the variables of a serial connection is equivalent to the global distribution for the variables of a divergent connection by just using definition 2.1.0.1:

$$P(X_i)P(X_j|X_i)P(X_k|X_j) = P(X_j, X_i)P(X_k|X_j) = P(X_i|X_j)P(X_j)P(X_k|Xj)$$

Definition 2.2.0.7. *A convergent connection $X_i \rightarrow X_k \leftarrow X_h$ is called a **V-Structure**, when there is no edge between X_i and X_h.*

Definition 2.2.0.8 (Equivalence)**.** *Two DAGs are equivalent, if they have the same underlying undirected graph and the same V-Structures.*

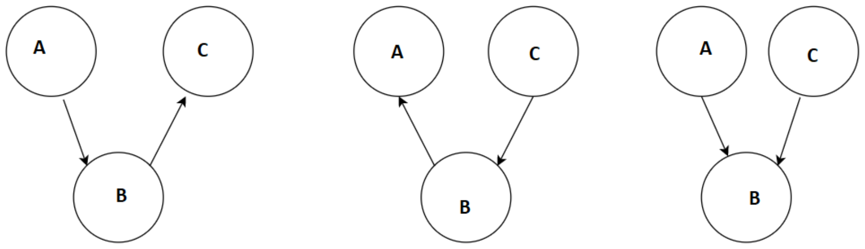


Figure 2.2: Three different structures

In figure 2.2, the first two structures do not contain V-structures and have the same underlying undirected graph, while the third structure contains a V-structure. So only the first two structures are equivalent. We can display equivalent structures in a so-called Partially Directed Acyclic Graph (PDAG), leaving the arcs that are not part of the common v-structure directed and the remaining arcs undirected. If we would be interested in the PDAG of the two equivalent structures of figure 2.2, we end up with the PDAG of figure 2.3

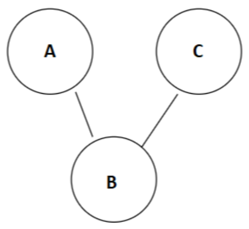


Figure 2.3: PDAG of the two equivalent structures of figure 2.2

Often, we are only interested in the behavior of one particular node. The Markov blanket of a node is the only knowledge that we need to predict the behavior of that node and its children.

Definition 2.2.0.9. *The* ***Markov Blanket*** *of a node* $A \in V$ *of a DAG* G *is the minimal subset of nodes* S *of* V *such that* $A \perp_G V \setminus (S \cup A) \mid S$

We can derive from theorems 2.2.0.5 and 2.2.0.4 the following result:

Theorem 2.2.0.10. *The Markov Blanket for every node of the graph of a BN consists of the parents, the children and all the other parents of the children.*

For example, in figure 2.4 we see an example of a Bayesian Network with all the nodes that belong to the Markov Blanket of A located in the grey circle. It is clear that all the parents, children and parents of the children are in the Markov Blanket.

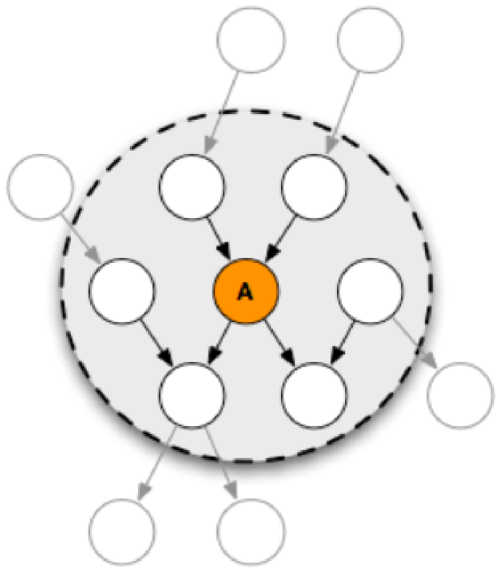


Figure 2.4: Markov Blanket for node A

2.3 Structure learning of Bayesian networks

To build up Bayesian networks for a given number of random variables, it is often not possible to rely solely on expert knowledge. It is often necessary to establish these Bayesian networks on the basis of data. There are different approaches to finding the most appropriate structure. You can divide these types of algorithms into two groups. Constraint-Based algorithms and Score-Based algorithms. Constraint-Based algorithms are based on the work of Pearl [3]. They try to find out the DAG of a Bayesian network through conditionally independence tests. During this research, no attention is paid to Constraint-Based algorithms.

You may be wondering why you cannot simply view each network individually and from there determine to what extent each network matches the given data. The problem to this is that the number of DAGS increases exponentially as the number of nodes increases. Robinson[4] showed that the number of unlabeled DAGs f(n) on n nodes can be expressed in the following way:

$$f(1) = 1 \tag{2.4}$$

$$f(n) = \sum_{i=1}^{n} (-1)^{i+1} \frac{n!}{(n-i)!i!} 2^{i(n-i)} f(n-1) \tag{2.5}$$

For example, when we want to find the DAG for 10 variables, we could choose among $4.2 * 10^{18}$ unlabelled DAGs.

2.3.1 Search procedures

As said above, we will only focus on score-based algorithms. Score-based algorithms consist of two components: a search procedure and a score function. Score-based algorithms are able to assign a score (more information on scores can be found in section 2.3.2) to a specific network and from this point try to adapt the network in such a way that the score will improve. A number of smart procedures can be used for these purposes: Hill Climbing and Tabu Search.

But how do we adapt the network? We start with a random structure. From this structure, we adjust the network in one of the following ways: we reverse an edge, we remove an edge or we add an edge. We then assign one of the scores to the adjusted networks. All the possible networks that could be made from one network are called the neighbors of a network. The next network you choose is the best-scoring network from the neighbors. You repeat this process. The danger here is that your starting structure can cause that you end up with a structure that gives us a local optimum.

In order to prevent this issue, we let the Hill-Climbing algorithm [5] start from multiple random positions to determine the network with the best score. The HC-algorithm shown below does take these steps into account

```
Data: Data set
Result: Network with the best score
x = Initial structure
MaxScore = Score(x)
while MaxScore increased in the previous step do
    for $s \in Neighbors(x)$ do
        if Score(s) > MaxScore then
            x = s;
            MaxScore = Score(s);
        end
    end
end
```

Algorithm 1: Hill-Climbing Algorithm

The Tabu-Search [6] algorithm, on the other hand, has less issues with local optima. This algorithm is structured as follows:

```
Data: Data set
Result: Network with the best score
BestStructure = initial structure
BestCandidate = initial structure
TabuList = [BestStructure]
while stoppingcondition() do
    for $candidates \in Neighbors(BestCandidate)$ do
        if $candidate \notin TabuList, Score(candidate) > Score(BestCandidate)$ then
            BestCandidate = Candidate
        end
    end
    if Score(BestCandidate) > Score(BestStructure) then
        BestStructure = BestCandidate
    end
    TabuList.add(BestCandidate)
end
```

Algorithm 2: Tabu-Search Algorithm

2.3.2 Score functions

In this section there will be an introduction to various types of score metrics which we can assign to a network. The following scores will be covered: Bayesian Information Criterion (BIC) and Akaike Information Criterion (AIC). The principle of assigning a score comes down to the following expression:

$$P(S, \theta_S \mid D) = \frac{P(D \mid S, \theta_S)P(\theta_S \mid S)P(S)}{P(D)} \tag{2.6}$$

where S stands for a specific BN, θ_S the parameters which belong to S and D for the data set.

AIC-Score:

The AIC-score can be calculated as follows:

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

where k is the number of dimensions and $\hat{L}$ is the likelihood function (see section 3.2).

So the method rewards the goodness of fit, and it penalizes the number of parameters, since a large number of parameters for a model always improves the goodness of fit. The aim is to find the network or model with the lowest AIC-value.

BIC-score:

The BIC-score can be calculated as follows:

$$BIC = \ln(N)k - 2\ln(\hat{L}) \tag{2.8}$$

where N is the number of data records, k the number of variables and $\hat{L}$ the maximized value of the likelihood function.

The BIC-Score also penalizes the number of parameters. This score can only be used if N is much larger than k. It is closely related to the AIC-score. The exact derivation of this score is studied in [7].

Both the BIC and the AIC score can be used in combination with the HC-algorithm or Tabu- algorithm.

2.4 R-package bnlearn

There is an R package called bnlearn [8] for learning a Bayesian network from data. The package was founded in 2007 by Marco Scutari. There are several algorithms and search-procedures implemented in bnlearn. The Hill-climbing and Tabu-search procedures are both implemented in this package. The package is able to deal with continuous, discrete and hybrid data. For this research, this package is used to determine the networks by different algorithms and to compare the predictive power of different obtained networks.

Chapter 3

Discrete Bayesian Networks

If we want to establish discrete Bayesian networks, the question is whether the given data is continuous or discrete. To determine this, it is important to be able to determine a certain threshold to check whether a random variable is continuous or not. This choice is generally quite suggestive. If a variable is nevertheless modelled as continuous, it does not mean that you cannot apply discrete Bayesian networks to it. Before we can apply Discrete Bayesian networks, we need to discretize the data set. One of the possible ways to define discretization is the following:

Definition 3.0.0.1. *A discretization of a real valued vector $\boldsymbol{x}$ of length N is simply an integer-valued vector* $\mathbf{d}$ *of identical length that satisfies the following properties [9]:*

- *each element of* $\mathbf{d}$ *is in the set {1,....,D-1} for some positive integer D*
- *for all i, j we have $d_i \leq d_j$ if and only if $x_i \leq x_j$*

As said above, the continuous data must be converted to discrete data before we can apply the concept of discrete Bayesian Networks. This is done through the use of different clustering (discretization) algorithms. The question which now probably arises is how to choose the right number of clusters. In order to choose the right number, we could make use of the silhouette statistic [10]. This statistic works as follows: Let

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i, j), \tag{3.1}$$

$$b(i) = \min_{i \neq j} \frac{1}{|C_i|} \sum_{j \in C_j} d(i, j). \tag{3.2}$$

where $|C_i|$ is the number of elements in cluster i and $d(i, j)$ is the Euclidean distance between clusters i and j.

The value b(i) gives us the smallest average distance of i to all points in any other cluster, while the value a(i) gives us the average distance between a point i and all the other points which are assigned to the same cluster. We need to compare the s(i)-values which can be computed by the following expression:

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}} \tag{3.3}$$

To find the right value for the cluster-numbers, we need to compare the average s(i)-value for the different groupings (with different number of clusters) which result from a clustering algorithm . The average value of s(i) is a measure to show how tightly the points are grouped.

Before we can determine the right number of cluster, we must first be able to discretize the data using some algorithms. In the next section such discretization algorithms will be discussed.

3.1 Discretization algorithms

K-means clustering by Hartiga-Wong method

The procedure [11] starts by determining the right number of clusters k. You then assign each point of the data to a random cluster $\{C_i\}_{i \in \{1,\dots k\}}$. After doing this, the method seeks to find the minimum of the following expression:

$$f(x,n,m) = \sum_{x \in C_m} (x-\mu_m)^2 + \sum_{x \in C_n} (x-\mu_n)^2 - \sum_{x \in C_m \setminus x} (x-\mu_m)^2 - \sum_{x \in C_n \bigcup x} (x-\mu_n)^2. \tag{3.4}$$

where μ_i is the mean of cluster i.

Once we have found the values x, n and m; we should remove x from C_n and move this point to C_m. The procedure terminates when equation 3.4 is positive for all the possible clusters and points. To better understand the method and the computational aspects of the method, I would like to refer you to [11].

Quantile discretization

The method aims to place the same number of elements in each cluster. This is done by sorting the observations that belong to a specific variable. Thereafter, the observations will be simply discretized by their ranking. For example, when we would like to discretize a variable $\mathbf{X}$ from a data set with 1000 observations into 10 levels. We will first use an efficient sorting algorithm, before we cluster every 100 subsequent observations of the sorted $\hat{X}$ in a different cluster. Thereafter, we should return the discretized values to their original positions.

Interval discretization

This method is not based on the ranking of the observations, but on their bin number in a histogram with D bins. When we have N observations $x_0, ...x_{N-1}$ we divide the interval between the first point x_0 and the last point x_{N-1} into D sub-intervals and then discretize according to the sub-interval an observation belongs. So the i^{th} observation belongs to level j if and only if

$$x_0 + \frac{j(x_{N-1} - x_0)}{D} < x_i < x_0 + \frac{(j+1)(x_{N-1} - x_0)}{D}$$

So we can use all of these discretization methods to convert a continuous data set into a discrete one, such that we will be able to apply discrete Bayesian Networks.

3.2 Parameter learning

As we saw earlier, we can derive the joint probability distribution of a multivariate random variable $\mathbf{X}$ from a network. The joint probability distribution looks as follow:

$$P(\mathbf{X}) = \prod_{i=1} P(X_i | Parents(X_i))$$

If all the variables are discrete, we use a Condition Probability Table, often abbreviated to CPT, to quantify the influence a parent has on his/her child. Parameter learning[12] is about using different techniques to estimate the values in such a table. There are some techniques that are based on a complete data set without missing values, while other algorithms aim to handle data sets with lack of information.

Parameter learning for complete data

Maximum likelihood estimation

Maximum likelihood estimation [13] is a method which determines estimations of parameters of a specific model. These values are chosen in a way such that the data that is observed is most likely to be produced by the model with these values as parameters. So for a specific Bayesian Network, which can be seen as a model, we need to estimate their parameters. But how do these parameters look like? When the data is discrete and there's no other information about the distribution of the variables from our data set, we are forced to estimate the terms of equation 2.3. Since the data is discrete, you should think of an estimation of the probability value of one of the value of a specific node given some combination of the possible values that the parents may have. The only way to estimate these probability values is by computing the relative frequency in our data set for these values.

Assume the data records are independent given all the estimated probability values θ and assume that the records are identically distributed, then we are able to express the likelihood function for a Bayesian network as follows:

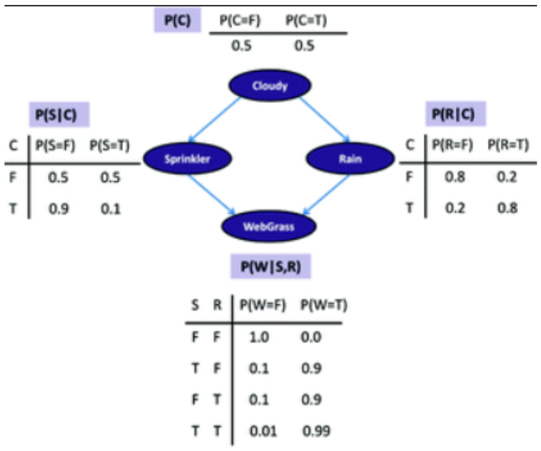
$$\mathcal{L}(\theta : D) = \prod_m P(x_{1m}, ...x_{nm}|\theta) = \prod_m \prod_i P(x_{im}|Parents(i), \theta_i) = \prod_i L_i(\theta_i : D) \tag{3.5}$$

where D is the data set of m records.

The maximum likelihood estimation θ^* is in fact a vector such that

$$L(\theta^* : D) = \sup_\theta L(\theta : D)$$

If we look thoroughly at figure 3.1, we see an example of a Bayesian Network where the the binary variables Sprinkler, Rain, Wetgrass and Cloudy are considered. What stands out is that a sprinkler and the occurrence of rain influence the wetness of the grass. This corresponds to reality. We can read our θ from this figure easily. For example, the parents of node w have four different configurations, because of the size of its table. ${\theta^*_{ij}}^k$ gives us the the estimation that node i has value j given the parents of node i have configuration k. We can estimate the parameter ${\theta^*_{ij}}^k$ by the number of records in our data set with the values of configuration k and records with value i for variable w divided by how many times configuration k occurs. So our likelihood estimation for the parameter consists of the set of all possible ${\theta^*_{ij}}^k$ for different values of i,j and k.



P(C)

P(C=F)	P(C=T)
0.5	0.5

P(S|C)

C	P(S=F)	P(S=T)
F	0.5	0.5
T	0.9	0.1

P(R|C)

C	P(R=F)	P(R=T)
F	0.8	0.2
T	0.2	0.8

P(W|S,R)

S	R	P(W=F)	P(W=T)
F	F	1.0	0.0
T	F	0.1	0.9
F	T	0.1	0.9
T	T	0.01	0.99

Figure 3.1: Conditional probability tables for 4 variables [5].

Bayesian estimation

Given a Bayesian network with unknown parameter θ and a complete data set, where θ is modelled as a random variable [14]. The goal is to determine

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)} \tag{3.6}$$

where $P(\theta)$ is called a prior distribution and $P(\theta|D$ is called the posterior distribution. The goal of the method is to compute the posterior density. Often, we make the assumption that the the prior distribution is the Dirichlet distribution [15].

Chapter 4

Gaussian Bayesian Networks

4.1 Multivariate normal distribution

It is well known that the normal distributed random variable X has the following probability density function

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma}} \exp^{-\frac{1}{2\sigma^2}(x-\mu)^2}$$

We can generalize this definition for a higher dimensional data set, assuming that every linear combination of variables follows a normal distribution. This is also known as the multivariate Gaussian distribution [16].

$$f_X(x = (x_1, ..., x_n)) = \frac{1}{(2\pi)^{n/2}|\sum|^{1/2}} \exp^{-\frac{(x-\mu)^T \sum^{-1}(x-\mu)}{2}} \tag{4.1}$$

where $|\sum|$ is the determinant of the covariance matrix and μ is the vector of means of the multivariate random variable X.

A Gaussian Bayesian network satisfies the following properties: all variables are continuous and multivariate normal.

Definition 4.1.0.1. *Suppose we have the continuous random variable Y. Y has a* ***Linear Gaussian Model*** *if there exist some continuous random variables $X_1, X_2, ...X_k$ such that*

$$P(Y|X_1, ..., X_k) = \mathcal{N}((B_0 + B_1\mu_1 + ... + B_k\mu_k), \sigma_Y^2)$$

where σ_Y^2 is the variance of variable Y, μ_i is the expectation of variable i and $B = (B_0, .., B_k)$ is a vector of constants.

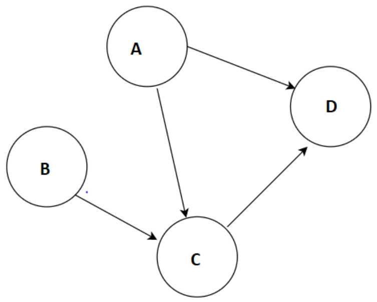


Figure 4.1: A network of 4 variables

For example, from figure 4.1 we can deduce the following terms which describe the joint probability distribution that belongs to the network:

$$\begin{aligned} P(B) &= \mathcal{N}(\mu_B, \sigma_B^2) \\ P(A) &= \mathcal{N}(\mu_A, \sigma_A^2) \\ P(C) &= \mathcal{N}(A_0 + A_1\mu_B + A_2\mu_A, \sigma_C^2) \\ P(D) &= \mathcal{N}(B_0 + B_1\mu_A + B_2\mu_C, \sigma_D^2) \end{aligned}$$

The parents of a certain node are seen as explanatory variables in terms of regression analysis. The parameters $\mathbf{B} = (B_0, ..., B_k)$ must be chosen from our data set with n variables in such a way that $\sum_{i=1}^{n}(\mathbf{B}^T X_i - Y_i)^2$ is minimal. The parameters B can be calculated by

$$\mathbf{B} = (A^T A)^{-1} A^T Y \tag{4.2}$$

where A is matrix that consists of the data that belong to the parents of a particular node and Y is the data column of that node.

4.2 Multivariate normality tests

In order to use Gaussian Bayesian Networks, we need to check whether the the variables of our data set are multivariate normally distributed. This is the case, when every linear combination of the variables is univariate normally distributed for some mean and variance. Two measures to asses univariate normality are the kurtosis(γ_1) and skewness(γ_2). The kurtosis(γ_1) is a measure for the tailedness and the skewness(γ_2) is a measure for the asymmetry of the distribution of a variable.

$$\gamma_1 = \frac{E[(X-\mu)^3]}{\sigma^3} \tag{4.3}$$

$$\gamma_2 = \frac{E[(X-\mu)^4]}{\sigma^4} \tag{4.4}$$

where μ is the expectation and σ is the standard deviation.

Assume we have a data set that consists of N records and k variables. The goal is to test whether these variables follow a multivariate normal distribution. One of the tests that can be applied to check if the assumptions are met is the so-called Mardia's test[17]. The below methods are all implemented in the MVN-Package [18] in R.

Mardia's test

You can imagine that if we are dealing with multivariate normality assumptions, we must have a measure for the skewness (asymmetry) and the kurtosis (tailedness) for all the linear combinations than can be derived from multivariate random variable. The most common test-statistic to asses the skewness (asymmetry) for a multivariate random variable is defined as follows

$$B_1 = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} g_{ij}^3 \tag{4.5}$$

and the test-statistic to asses kurtosis for a random vector is defined as

$$B_2 = \frac{1}{N}\sum_{i=1}^{N} g_{ii}^2 \tag{4.6}$$

where $g_{ij} = (x_i - \overline{x})^T A^{-1}(x_j - \overline{x})$, $A = \frac{1}{N}\sum_i (x_i - \overline{x})(x_i - \overline{x})^T$ and $\overline{x} = \frac{1}{N}x_i$.

With the help of these two test statistics we see that

$$z_1 = \frac{(k+1)(N+1)(N+3)}{6(N+1)(k+1)-36} B_2 \tag{4.7}$$

is approximately χ^2 distributed with k(k+1)(k+2)/6 degrees of freedom and

$$z_2 = \frac{b_{2,k} - k(k+2)}{\sqrt{\frac{8k(k+2)}{N}}} B_1 \tag{4.8}$$

is approximately standard normally distributed.

Henze-Zirkler's test

If the data are distributed as multivariate normal, the test-statistic in (4.9) is approximately log-normally distributed, with some mean and variance that will follow below. The test statistic of Henze-Zirkler[19] is defined as follows:

$$HZ = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n}\exp^{-\frac{\beta^2}{2}D_{1j}} - 2(1+\beta^2)^{-\frac{k}{2}}\sum_{i=1}^{n}\exp^{-\frac{\beta^2}{2(1+\beta^2)}g_i} + n(1+2\beta^2)^{-k/2} \tag{4.9}$$

where $\beta = \frac{1}{\sqrt{2}}(\frac{n(2k+1)}{4})^{\frac{1}{k+4}}$ and $D_{ij} = (x_i - x_j)^T A^{-1}(x_i - x_j)$. From this expression, because we know that this statistic is log normally distributed it is possible to express the mean and the variance of the statistic.

$$\mu = 1 - \frac{a^{-\frac{k}{2}}(1 + k\beta^{\frac{2}{a}} + (p(p+2)\beta^4))}{2a^2}$$

$$\sigma^2 = 2(1+4\beta^2)^{-\frac{k}{2}} + \frac{2a^{-k}(1+2k\beta^4}{a^2}) + \frac{3k(k+2)\beta^8}{4a^4} - 4w^{-\frac{k}{2}}(1 + \frac{3k\beta^4}{2w} + \frac{k(k+2)\beta^8}{2w^2})$$

where $a = (1+2\beta^2)$ and $w = (1+\beta^2)(1+3\beta^2)$, hence the log-normalized mean and log-normalized variance are $\log(\mu) = \log(\sqrt{\frac{\mu^4}{\sigma^2+\mu^2}})$ and $log(\sigma^2) = \log(\frac{\sigma^2+\mu^2}{\sigma^2})$.

The Wald test-statistic to asses multivariate normality is defined as

$$z = \frac{\log(HZ) - \log(\mu)}{\log(\sigma)} \tag{4.10}$$

Royston's MVN test

Royston's test[20] makes use of the Shapiro-Wilk test to test that a sample $x_1, .., x_N$ can from an (univariate) normally distributed population.

$$W = \frac{(\sum_{i=1}^{n} a_i x(i))^2}{\sum_{i=1}^{n}(x_i - \overline{x})^2}$$

where $x(i)$ is the ith smallest number in $x_1, ...x_n$ and the a_i can be computed by the following expression: $(a_1, ..., a_n) = \frac{m^T V^{-1}}{C}$ where $C = (m^T V^{-1} V^{-1} M)^{1/2}$, m is made of the expected values of the ordered variables when making n independent draws and V is the covariance matrix of these ordererd variables.

Royston's test also makes use of the Shapiro-Francia test

$$W = \frac{cov(x, m)}{\sigma_x \sigma_m} \tag{4.11}$$

If the kurtosis is greater than 3, then the Royston test uses the Shapiro-Francia test, otheriwse it uses the Shapiro Wilk test. Let W_j the Shapiro-Wilk or Shapiro-Francia test and Z_j the values obtained from a normality transformation.

When $4 \leq N \leq 11$ use $x = N$ and $w_j = -\log(\gamma - \log(i - W_j))$ and when $12 \leq N \leq 2000$ use $x = \log(N)$ and $w_j = \log(1 - W_j)$.

The transformed variables Z_j can be computed by

$$Z_j = \frac{w_j - \mu}{\sigma}$$

where γ, μ and σ are derived by the polynomials below:

$$\gamma = a_{0\gamma} + a_{1\gamma}x + ... + a_{d\gamma}x^d$$
$$\mu = a_{0\mu} + a_{1\mu}x + ... + a_{d\mu}x^d$$
$$\sigma = a_{0\sigma} + a_{1\sigma}x + ... + a_{d\sigma}x^d$$

We have now finally arrived at the point that makes us able to formulate the Royston's test statistic

$$H = \frac{e\sum_{j=1}^{k}\psi_j}{k} \tag{4.12}$$

This statistic is approximately χ^2 distributed with e degrees of freedom and Φ is the cumulative distribution function for the standard normal distribution such that

$$e = \frac{k}{1 + (k-1)\overline{c}}$$
$$\psi_j = (\Phi^{-1}(\Phi(-Z_j)/2))^2$$

The term $\overline{c}$ also needs to be calculated

$$\overline{c} = \sum_i \sum_j \frac{c_{ij}}{k(k-1)}$$

where r_{ij} is the correlation between the i^{th} and j^{th} variables.

$$c_{ij} = \begin{cases} g(r_{ij}, n) & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}$$

and where $g(0, n) = 0$ and $g(1, n) = 1$ and g is defined as follows:

$$g(r, n) = r^{\lambda}(1 - \frac{\mu}{v}(1 - r)^{\mu})$$

The parameters λ, μ and v were estimated from a simulation study by Ross [21].

He found that the values $\mu = 0.715$ and $\lambda = 5$ and v can be found by

$$v = 0.21364 + 0.015124\log(n)^2 - 0.0018034\log(n)^3$$

Chapter 5

Study Case: Canadian researchers

Bibliometrics [22] is the science that revolves around analyzing publications and their citations using statistical methods. Methods from bibliometrics are mainly used in information sciences. It is often tried to show how much influence a particular publication or researcher has had. Bayesian networks are used in this research to perform this analysis.

5.1 Data description

In order to understand which factors and to what extent these factors play a role in the results achieved by a particular researcher, it is essential that bibliometric data is studied. In this section we are going to look at a data set that comes from the Web of Science (WoS). Initially, the data set consisted of 13626 records. This data set has been adapted by removing researchers born before 1959. Only researchers who obtained their PhD-degree after 1980 are also taken into account. Older academics in the database who published before 1980 do not have an entry in the year of first publication, therefore we omit anybody who has a birth year earlier than 1959 (under the assumption that nobody will have published before they are 20 years old). Subsequently, all researchers were removed who made their first publication after 2010, due to the fact that such a researcher is too short a time active to draw conclusions about his/her performance. We ended up with a data set with 1338 records and 14 variables. This data set has already been used by various researchers for their bibliometric research [23]. In the table below you can see where every variable of the data set stands for:

Variable	**Description**
pp_top_prop	Proportion of publications in top 10% most cited publications in their field
fpy	Year of the first publication in the WoS
authors_paper	Average number of authors per paper of the scholar
countries_paper	Average number of countries per paper of the scholar
refs_paper	Average number of references per paper of the scholar
pp_pos_1	Proportion of publications with the scholar in the first place
pp_pos_last	Proportion of publications with the scholar in the last place
p	Number of publications of the scholar on WoS
Journal Score	Journal citation indicators, determining the percentage of publications in the journal that are in the top 10% most cited publications in their field.
pp_collab	The percentage of publication that are collaborative
pp_int_collab	The percentage of publications that result from an international collaboration
pp_industry	Proportion of publications with any type of indistrual collaboration produced by the scholars
bithyear	Birthyear
Phd_year	The year of obtaining PhD-degree.

Table 5.1: Data description

You can take a look at figure 5.1 to get an impression of the distributions for the different variables. It becomes clear from this figure that only the variables fpy, pp_pos_last, pp_collab and refs_paper look like some normal distribution.

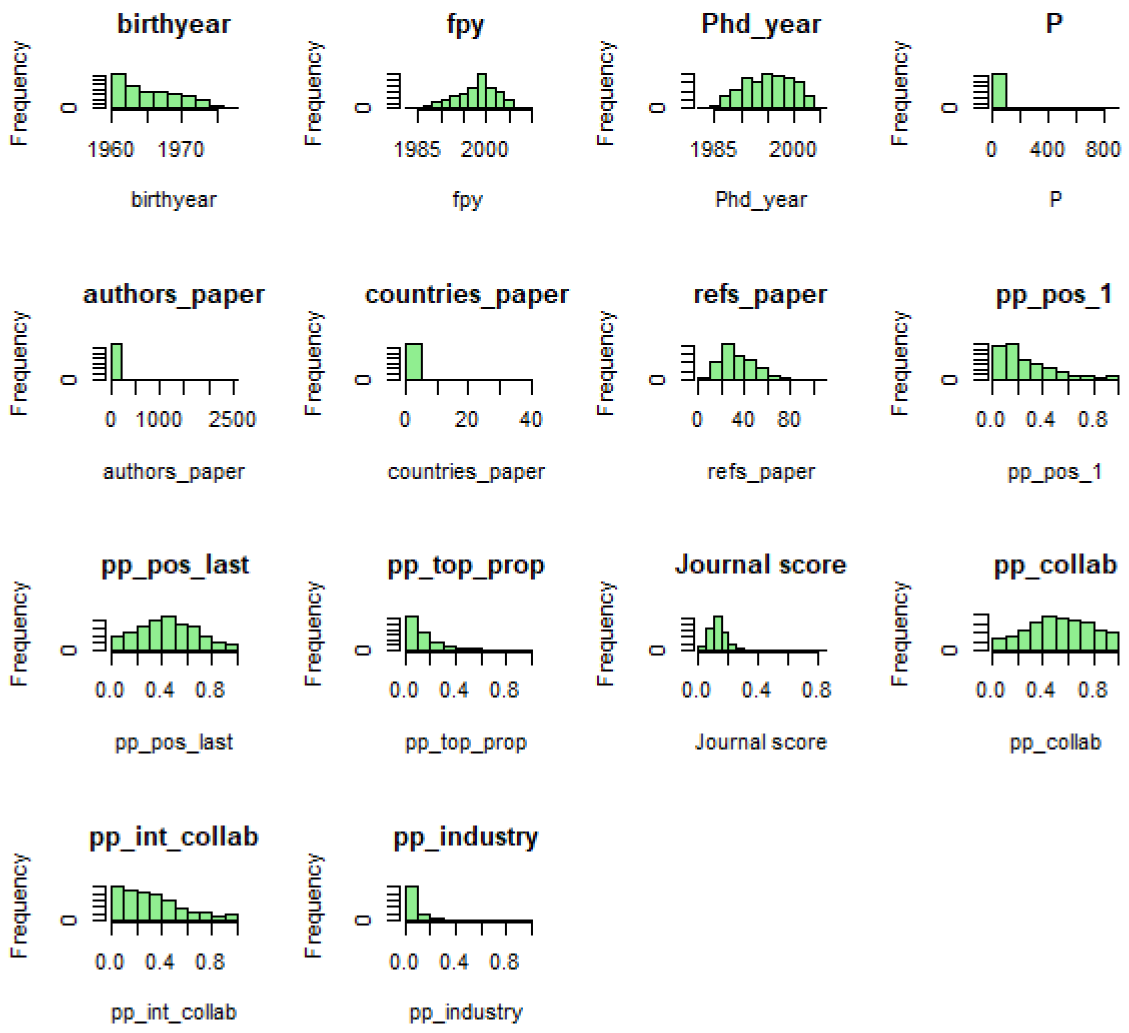


Figure 5.1: Histograms of the variables of our data set

To get an impression of how different variables correlate, we will make use of the Pearson correlation coefficient to measure this correlation. The coefficient must be calculated in the following way:

$$\rho(X,Y) = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \tag{5.1}$$

The values of these coefficients lie between -1 and 1. The closer to 1, the more the two variables correlate positively. If the value is close to 0, then there is no relation between the two variables and if the value is close to -1 there is a negative relation between them. We can calculate these coefficients for every combination of two variables of our data set and can put it in a matrix. See figure 5.2 for the correlation matrix of the bibliometric data set.

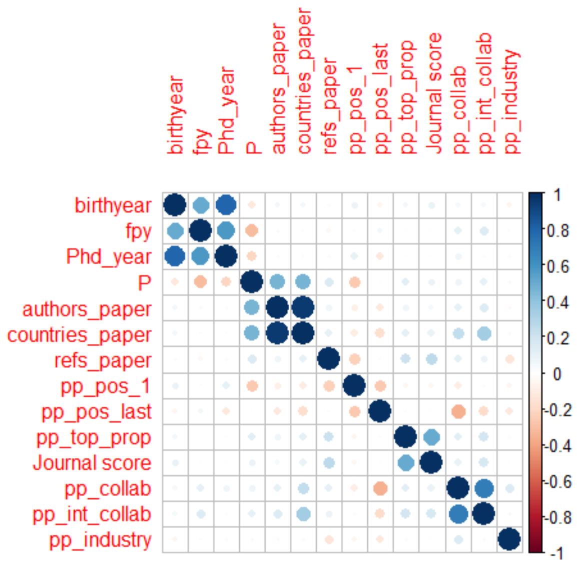


Figure 5.2: Correlation plot to see correlations between variables

The correlation between birthyear, Phd_year and fpy immediately stands out. The high number of variables that have some kind of a relation with P is also worth mentioning.

In this thesis we will focus primarily on the variable pp_top_prop, which we will use as a measure of performance. We are interested in predicting this variable. So this variable will be our target variable.

5.2 k-fold cross validation

Cross-Validation [24] is a technique which is used to asses how a particular model will perform in terms of prediction given another independent data set with the same variables. For example, it can be used to check how a Bayesian network will predict a target-variable, when it is exposed to unseen data. This is done by splitting the original data set into a training part and a test-part. The parameters of the model are estimated by making use of the training-part. Thereafter the test-set is used to check how "good" the prediction of the Bayesian network will be with the estimated parameters. One of the metrics that can be used to determine this "goodness" is the Mean Squared Error (MSE). MSE takes the sum of the squares of the differences between the observed values and the predicted values into consideration.

$$MSE = \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2 \tag{5.2}$$

One of the most common cross-validation procedures in Bayesian network is the k-fold cross validation. This procedure works in the following way. The original data set is partitioned into k equal samples. One of the k samples is chosen to be the test set, and the remaining sets will be used as training set. This process is repeated k times, with each of the k samples used once as the test set. So all our records of our data set will be used for both test and training. Although k is an unfixed parameter, 10-fold cross-validation is commonly used.

5.3 Bootstrap procedure

Suppose we have a data set D of N records and we have a Bayesian Network G(D) that is returned by one of the data-driven algorithms that are defined in section 2.3. We also assume that every record of our data set is independent of the Bayesian Network. Suppose now we are interested in the existence of a specific arc in the network that will be returned by our algorithm. We denote the occurrence of this feature by a Boolean f. To estimate the confidence that this feature will happen, we can define the following quantity:

$$P_N(f) = P(f(G(D) = 1||D| = N)$$

This gives us the probability that the feature f will occur given a data set of N records that is sampled with replacement from our original data set. Since we only have one data set we need to use such a sampling technique called bootstrap. In this research, we focus solely on non-parametric bootstraps. It is about getting confidence on the existence of a feature, even if the data set is different from the original data set. The non-parametric bootstrap-method [25] works as follows: make a data set with N records by sampling with replacement. Then you need to apply a learning algorithm on this data set. Repeat this process m times and calculate the probability that the feature will occur by dividing the number of times the feature of interest occurred and the m steps. So $P_n(f)$ can be estimated by this simple process. Of course, we can apply this procedure to obtain some confidence about other features like whether a specific variable is in the Markov Blanket of another variable. However, in this research, we will focus only on the existence of a specific arc.

Because in this research we are mainly interested in how different networks will operate in terms of prediction on a data set that is completely unknown , we will use this non-parametric bootstrap method to build up networks. This means that we will only add a possible edge into our network if $P_N(f)$ is larger than a chosen α. So we will only add edges with confidence higher than α. The question that may arise now is, how high do we choose α. In the remainder of the report, it will become clear how we deal with this issue.

5.4 Results

In this section we will try to apply the previously described theory to our data set of researchers. In this section we will also try to build different networks that may provide a good MSE value. We will use 10-fold cross validation to validate. The reason to choose 10 for the parameter k, is because 10 is commonly used in practice.

5.4.1 Network Comparison for Discrete Bayesian Networks

Using the different cluster techniques that can be found in section 3.1 always leads to the network in figure 5.3. We can't do a lot of inference by making use of this network, since we can determine very few relationships between different variables. We only find edges from countries_paper to authors_paper and pp_int_collab. The edge from countries_paper to pp_int_collab makes sense, since the number of countries per paper is of course highly correlated with the percentage of publications that result from an international collaboration. The existence of the other edge is less obvious. On the other hand we can't find any variables in the Markov Blanket of our target variable pp_top_prop. So predicting the behavior of this variable will become hard. The silhouette statistic is used to determine the number of clusters for every variable. Using this test statistic, an attempt was made to subdivide each variable into 4,8,12,16 or 20 clusters, depending on the value of the statistic. You can imagine that you will get huge amounts of CPT values through the use of a discrete Bayesian network, so computationally it is very inefficient. One possible reason for failing to find a correct network is that the use of discretization has affected the dependency structure between different variables in such a way that building a network has become virtually impossible.

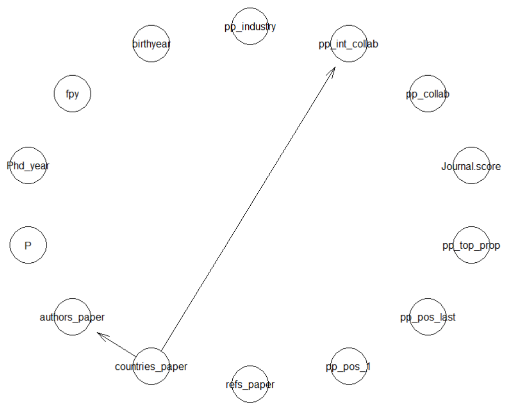


Figure 5.3: Network that is returned by the discrete-based algorithms

5.4.2 Network comparison Gaussian Bayesian Networks

Multivariate normality assumption

Before we start to predict, we will first examine the extent to which all our variables meet the multivariate normality assumption. For the HZ test we get the value 2.116773 and the p-value 0. We do not meet the assumption according to this test. The Mardia-test gives us for the skewness-value 109240.790751032 and for the kurtosis-value 613.2662014710069. The p values for both are measures 0. To confirm the feeling we already had, the Royston test also showed that we do not meet the assumption. The results for the different test can be found in the table below.

Test	**Statistic**	**p-value**	**Result**
Mardia skewness	109240.790751032	0	No
Mardia kurtosis	613.266204710069	0	No
Royston	1700.674	0	No
Henze-Zirkler	2.116773	0	No

A p-value of 0 means that the results are highly significant. It tells us that the probability to get our data set, if the null hypothesis (Multivariate normality) is true, is very low. Despite this result, we are nevertheless interested in how the Gaussian Bayesian Networks will predict our target-variable pp_top.

Predictive performance of different networks

First of all, an attempt was made to build a network through the use of the HC-algorithm. Because with this algorithm the starting network is very decisive for the result, we have run this algorithm 10 times with 10 different starting networks. The final chosen network is the network, with the BIC score being the best. With the help of bnlearn, this network gives us the network from figure 5.4. The BIC-score of this network is -27499.08. What we see is that there are no edges to our target variable pp_top_prop. We expect an MSE that will be approximately equivalent to the variance of the target variable, since the target variable will be predicted by the mean, because of the lack of explanatory variables. In our case, this is a disappointing result, since we could determine the same result by just using the mean of pp_top_prop to predict.

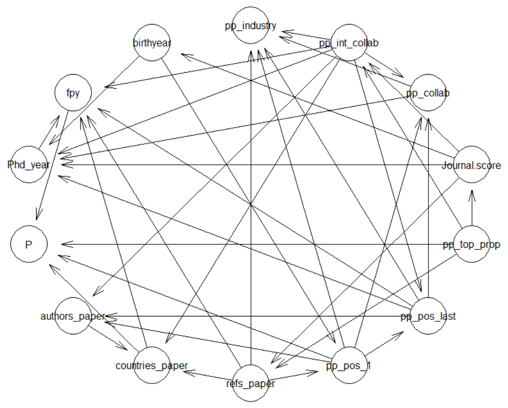


Figure 5.4: Network 1: Network produced by HC-algorithm

The Tabu-algorithm was then used to build a network. Within this concept, different iterations do not have to be used due to the nature of the algorithm. The network (see figure 5.5) that we get as a result is a network where different variables are parents for our target variable, namely P, refs_paper and Journal.score. The main difference with the HC-algorithm is that the edges between pp_top_prop and these variables are in the direction of pp_top_prop, while in the HC-algorithm there are edges between pp_top_prop and P, refs_paper and Journal Score, but these edges point in the opposite direction.

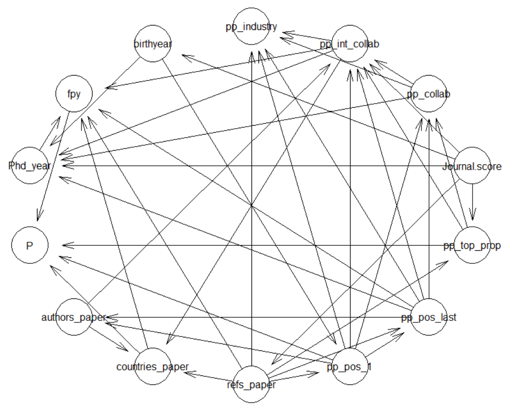


Figure 5.5: Network 2: Network produced by Tabu-algorithm

During the process of learning the right network, you can get a network that does not correspond to reality. The use of prior information, which is often based on logical reasoning, is possible in such cases to be added to the network in the form of blacklist and whitelist. In the blacklist you add all arcs that cannot possibly occur and in the whitelist you add all arcs that must occur. It can be a chore to manually add all possible arcs to the blacklist or whitelist. This will especially become an issue when the number of variables of a data set to which you want to apply a learning algorithm is huge. In such cases, only the arcs from an earlier obtained network are considered to add to our blacklist or whitelist. For the whitelist, we have chosen to add the following three edges:

Whitelist
Phd_year to P
Phd_year to fpy
birtyear to Phd_year

Table 5.2: Whitelist

The reason to add these tree edges, is because were are quite sure about their existence. The year of obtaining a PhD-degree has a lot of influence on the number of publications, because someone who has just obtained his or her degree has not had the time to publish. The reason to add the edge from PhD_year to fpy is because the year of obtaining PhD-degree will definitely have some influence on the year of first publication. Since the birtyear of a researcher is always less than the Phd_year, we expect that there is some relation between these variables. Of course, we could have add the edges from our Tabu-Network(see figure 5.5) that point in the direction of our target variable, but we have not done this yes, because we are not very sure about their existence. So we have chosen to only add the edges that we are 100 percent sure of. For the blacklist, we have chosen to add the following edges:

Blacklist
Phd_year to birthyear
P to birthyear
fpy to birtyear
pp_pos_last to birtyear
pp_int_collab to pp_collab
pp_int_collab to fpy
pp_pos_last to fpy
pp_pos_last to Phd_year
Journal score to refs_paper
P to pp_top_prop

Table 5.3: Blacklist

The choice to add all these edges is based on the following rule of thumb: If one variable certainly has no influence on the other variable, we will add the edge between these variables into to the blacklist. Again we did not choose to consider edges from the two previously obtained networks, because we are currently quite uncertain about which edges will not occur. Adding edges to the blacklist ensures that we exclude the possibility of existence.

It is now time to use the previously defined blacklist and whitelist that hopefully could ensure that we will get a good prediction by using the HC-algorithm. However, this hope is in vain, once again we are dealing with edges that do not go in the direction of our target variable (see figure 5.6).

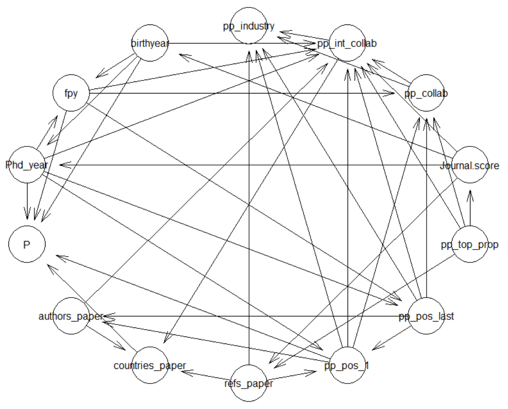


Figure 5.6: Network 3: Network produced by HC-algorithm with blacklist and whitelist

Although we got a nice result by using the Tabu algorithm, it can't hurt to check whether we might get better results by adding the blacklist and whitelist. The obtained network can be found in figure 5.7. There are two edges that point in the direction of our target variable; an edge from refs_paper and another edge from Journal Score. When we compare this network with the Tabu-network of 5.5, these two edges are the only edges that point in the direction of our target variable. It appears that adding the blacklist and whitelist gives us an MSE-value of 0.01289887, which is a bit lower than the MSE-value of 0.01283498 for the network in figure 5.5. This difference in value is due to the use of a numerical algorithm called the QR-algorithm [26] for estimating the parameters of our linear models.

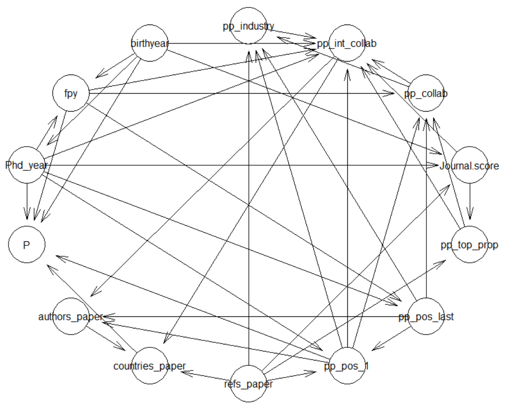


Figure 5.7: Network 4: Network produced by TABU-algorithm with blacklist and whitelist

Because we are also interested in which network we get when we apply the bootstrap-procedure with a certain threshold. Hopefully we can now get a good result from the HC-algorithm. We set the treshold at 0.75. The reason for choosing this value is because we have tried a number of different threshold values for the Tabu-algorithm and looked at the MSE-values for the different obtained networks. The results can be found in the following table:

Threshold value	**MSE**
0.5	0.01661752
0.6	0.01663301
0.7	0.01663196
0.8	0.01723760
0.9	0.01723760

Table 5.4: MSE-values for different thresholds

Despite the fact that the value for 0.5 is the best, we decide not to opt for this. It is of course true that if we keep the threshold value low, we get more edges in our network. So it can be very tempting to keep the threshold value very low, because then we have more chance of more edges in the direction of our target variable. However, we are not going to do this because we want to use the bootstrap procedure in the way it is intended. We have therefore chosen to set the threshold to 0.75. However, the resulting network for the bootstrap procedure mixed with the HC-algorithm is disappointing. There are no edges in the direction of our target variable(see figure 5.8).

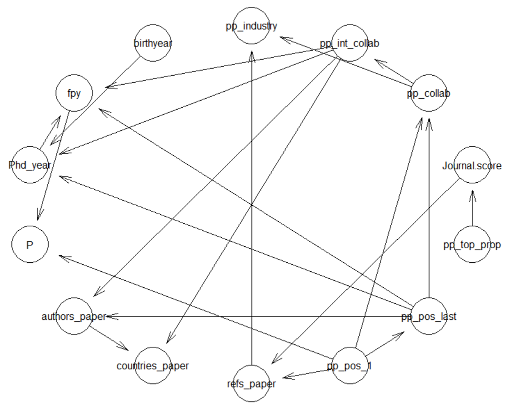


Figure 5.8: Network 5: Network prodcuced by the bootstrap-procedure and HC-algorithm

If we now continue to apply the blacklist and whitelist with the bootstrap procedure for our HC-algorithm, we will again get a network where no edges point towards our target variable (see figure 5.9).

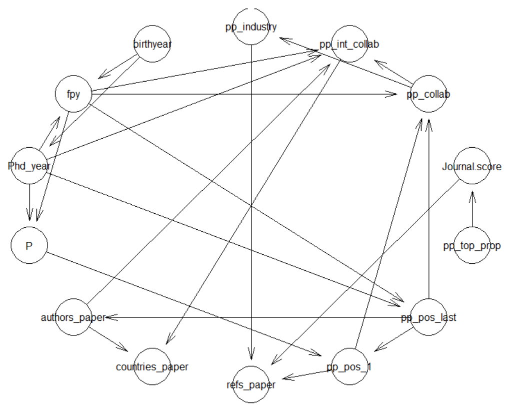


Figure 5.9: Network 6: Network prodcuced by the bootstrap-procedure, HC-algorithm, blacklist and whitelist

Although we already had some nice results for the Tabu algorithm, there is the hope that we can make it better by applying the bootstrap procedure. An interesting fact is that there is now an edge from pp_int_collab towards our target variable (see figure 5.10). Nevertheless, the MSE value is now 0.01663533. This value is slightly worse than that of the previous networks.

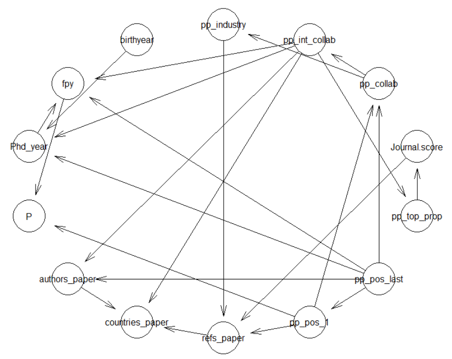


Figure 5.10: Network 7: Network prodcuced by the bootstrap-procedure and Tabu-algorithm

When we use the defined blacklist and whitelist together with the bootstrap procedure, wee see that there are no edges in the direction of our target variable (see figure 5.11).

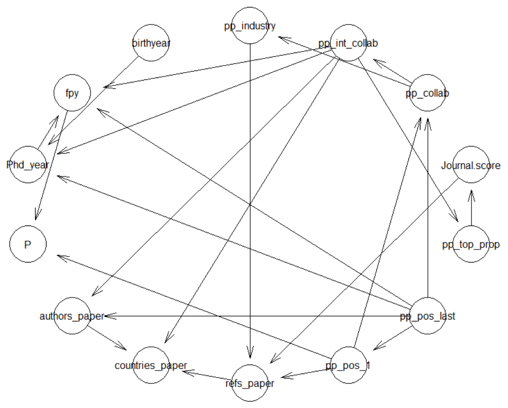


Figure 5.11: Network 8: Network produced by the bootstrap-procedure, Tabu-algorithm, blacklist and whitelist

What becomes very clear is that the Tabu algorithm and the HC-algorithm differ greatly in the direction that the edges have. It has often happened with the HC-algorithm that certain edges point away from our target variable, while the edges in the opposite direction are part of a network produced by the Tabu algorithm. A possible solution may be to whitelist all the edges of which our target variable has been a part, in the direction of our target variables. This means that we need to add edges from refs_paper, p, pp_int_collab and Journal towards our target variable pp_top_prop. The MSE-value for this new network is 0.01272587. This is the best value we have seen so far. The network can be found in figure 5.12.

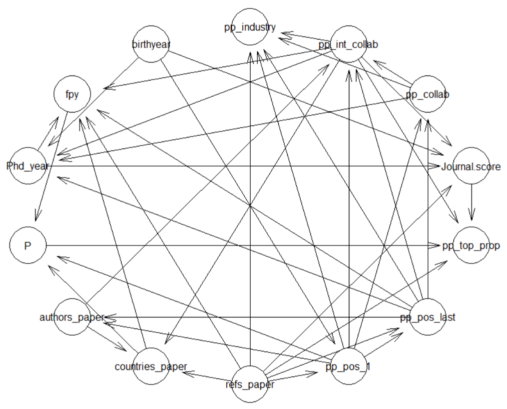


Figure 5.12: Network 9: Network produced by HC-algorithm with new defined whitelist

You may be wondering to what extent we can see from the BIC score the predictive power of a network. The following table shows that the BIC score certainly does not influence the prediction of pp_top_prop, since Network 9 has the best-scoring MSE-value and the second worst BIC-score.

Network	**Blacklist or whitelist**	**Bootstrap**	**MSE**	**BIC- score**	**# Arcs**
Network 1: HC	NO	NO	0.01719681	-27499.08	40
Network 2: Tabu	NO	NO	0.01283498	-27504.96	41
Network 3: HC	YES	NO	0.01723294	-27511.03	41
Network 4: Tabu	YES	NO	0.01289887	-27510.55	40
Network 5: Tabu	NO	YES	0.01721951	-27728.96	25
Network 6: Tabu	YES	YES	0.01720736	-27725.93	25
Network 7: HC	NO	YES	0.01663533	-27737.33	25
Network 8: HC	YES	YES	0.01722387	-27744.17	25
Network 9: Tabu	YES	NO	0.01272587	-27503.76	40

Table 5.5: Results for the different networks

5.4.3 Self-composed algorithm

During the construction of different networks using different algorithms, the idea arose to combine the theory of clustering with that of Gaussian Bayesian Networks. An algorithm has been attempted that uses different Bayesian networks for different inputs to make predictions.

First of all we start by choosing m variables, that best describe the data set. Use can be made of techniques such as PCA or factor analysis[27]. We then try to divide the data set into k parts, using the the Hartiga-wong method and the Euclidean Distance as the distance function. After the data set is divided into k parts, we run a learning algorithm on each part. So we will end up with k networks. If we want to apply prediction one by one for a new data set (with the same variables), we look at the Euclidean distance between the value of the variables of our data record and the average of each part of our original data set. We then choose the network that belongs to the shortest distance. We have chosen to split our data set into 5 parts and we have chosen the following 6 variables as main variables: birthyear, P, authors_paper, fpy and countries_paper. However, this choice for these values is a subjective one. As mentioned earlier, different techniques could be used to better determine these values.

As said above, the first step in the process is to cluster the data. The Hartiga-wong method given us the following means for each cluster:

Clusternumber	**fpy**	**Birthyear**	**Phd_year**	**P**	**authors_paper**	**countries_paper**
1	1965.298	1996.452	1994.474	55.72704	8.613078	1.610372
2	1963.583	1994.042	1992.222	139.12500	7.985599	1.637599
3	1966.250	1996.000	1995.250	349.00000	837.348143	16.185511
4	1966.539	1999.335	1996.585	16.98150	4.201006	1.450337
5	1970.750	1996.500	1998.500	288.75000	1963.107342	28.821237

Table 5.6: The mean for the main variables of the clusters

Because the values of the TABU algorithm were good compared to the HC-algorithm, we will only apply this self-composed algorithm in combination with the HC-algorithm. So we can now use the HC-algorithm for each cluster. We can find the network that is returned by the HC-algorithm in figure 5.13. There is an edge from birthyear to our target variable.

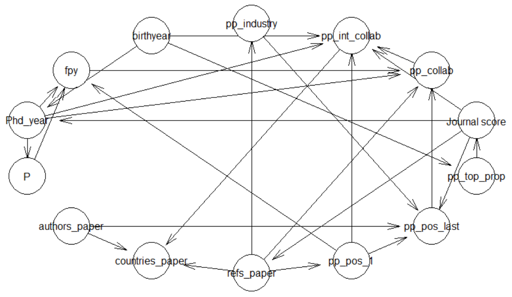


Figure 5.13: Network for cluster 1

The network that belongs to the second cluster has an edge from refs_paper to our target variable.

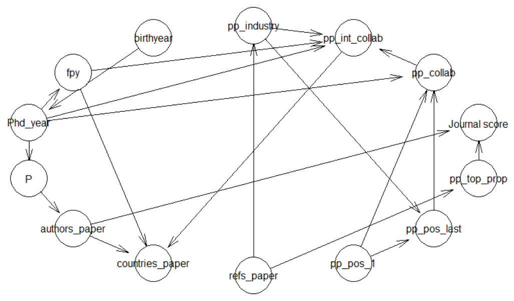


Figure 5.14: Network for cluster 2

The network (see figure 5.15) that belongs to our third cluster has two edges to our target variable: one from birthyear and one from refs_paper.

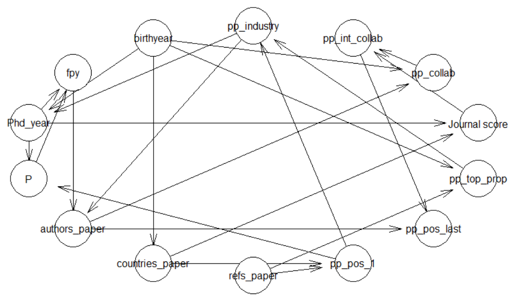


Figure 5.15: Network for cluster 3

The network (see figure 5.16) that belongs to our fourth cluster has one edge from countries_paper to our target variable.

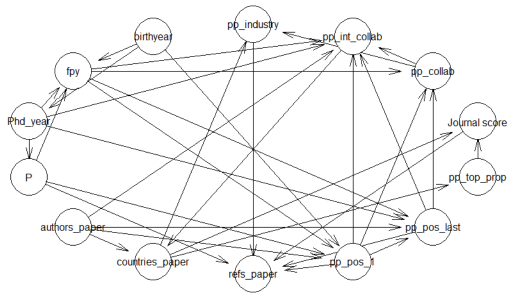


Figure 5.16: Network for cluster 4

The network (see figure 5.17) that belongs to our fifth cluster has no edges that point to our target variable.

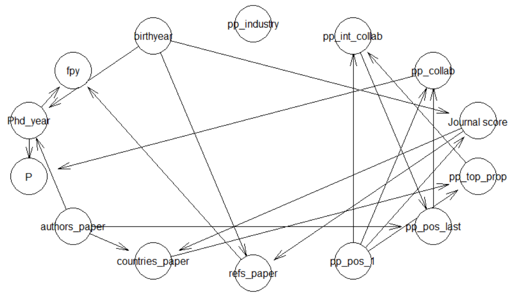


Figure 5.17: Network for cluster 5

So what we see is that with regard to edges in the direction of our target variable, we are making considerable progress with the HC- algorithm. We are now ready to start predicting.The MSE value that is returned by the use of these networks is 0.0143184. So we made some improvement for the HC-algorithm.

Chapter 6

Conclusions and recommendations

To start with, we have seen that there are many possibilities for learning a certain network. For example, we have seen that we can use different search procedures and also assign different scores to networks. This large amount of choices that could be made means that you cannot view every possible combination due to a lack of computational time.

In the beginning we had the idea to build networks by means of discretization. However, this proved difficult to achieve, because after discretization it appeared that the independencies between our variables virtually disappeared. In the future, for example, the Gaussian Mixture Model [28] could be taken into account to better manage this process.

We then switched to the Gaussian Bayesian Networks. We showed some nice results by using this model. The Tabu algorithm combined with the BIC-score gave us a nice MSE score. Apparently the self-made blacklists and whitelist even improved the structure that does not used these lists. For the HC-algorithm, the results were somewhat more disappointing. We found structures for this algorithm that did not contain edges pointing towards our target variable. Even though we added blacklists and whitelists, the result did not improve significantly.

In order to give the HC-algorithm a chance of success, we tried to apply our own algorithm, using clustering. This resulted in a significant improvement compared to what we had seen before. We can nevertheless make a few comments on this, since a number of parameters for this algorithm have been chosen on a subjective basis.

In the future, consideration could be given to applying constraint-based algorithms to the used data set. Perhaps this would produce even better results in terms of prediction. In addition, consideration could be given in the future to certain metrics that have been used. For example, apart from the BIC-score and the AIC-score, it is possible to involve other score metrics in our analysis. Finally, there is the possibility to consider the use of other search procedures like the Genetic algorithm.

Bibliography

[1] Daphne Koller, Nir Friedman, Sašo Džeroski, Charles Sutton, Andrew McCallum, Avi Pfeffer, Pieter Abbeel, Ming-Fai Wong, David Heckerman, Chris Meek, et al. *Introduction to statistical relational learning*. MIT press, 2007.

[2] Marco Scutari and Jean-Baptiste Denis. *Bayesian networks: with examples in R*. Chapman and Hall/CRC, 2014.

[3] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier, 2014.

[4] Robert W Robinson. Counting unlabeled acyclic digraphs. In *Combinatorial mathematics V*, pages 28–43. Springer, 1977.

[5] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

[6] Fred Glover. Tabu search—part i. *ORSA Journal on computing*, 1(3):190–206, 1989.

[7] Harish S Bhat and Nitesh Kumar. On the derivation of the bayesian information criterion. *School of Natural Sciences, University of California*, 2010.

[8] Marco Scutari. Learning bayesian networks with the bnlearn r package. *arXiv preprint arXiv:0908.3817*, 2009.

[9] Elena S Dimitrova, M Paola Vera Licona, John McGee, and Reinhard Laubenbacher. Discretization of time series data. *Journal of Computational Biology*, 17(6):853–868, 2010.

[10] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[11] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[12] Zhiwei Ji, Qibiao Xia, and Guanmin Meng. A review of parameter learning methods in bayesian network. In *International Conference on Intelligent Computing*, pages 3–12. Springer, 2015.

[13] John Aldrich et al. Ra fisher and the making of maximum likelihood 1912-1922. *Statistical science*, 12(3):162–176, 1997.

[14] James O Berger. *Statistical decision theory and Bayesian analysis.* Springer Science & Business Media, 2013.

[15] Thomas Minka. Estimating a dirichlet distribution, 2000.

[16] Yung Liang Tong. *The multivariate normal distribution.* Springer Science & Business Media, 2012.

[17] Kanti V Mardia. Measures of multivariate skewness and kurtosis with applications. *Biometrika*, 57(3):519–530, 1970.

[18] Selcuk Korkmaz, Dincer Goksuluk, and Gokmen Zararsiz. Mvn: An r package for assessing multivariate normality. *The R Journal*, 6(2):151–162, 2014.

[19] N Henze and B Zirkler. A class of invariant consistent tests for multivariate normality. *Communications in Statistics-Theory and Methods*, 19(10):3595–3617, 1990.

[20] Samuel Sanford Shapiro and Martin B Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591–611, 1965.

[21] GJS Ross, RD Jones, RA Kempton, FB Laukner, RW Payne, D Hawkins, and RB White. *MLP: maximum likelihood program.* 1980.

[22] Philipp Schaer. Applied informetrics for digital libraries: an overview of foundations, problems and current approaches. *Historical Social Research/Historische Sozialforschung*, pages 267–281, 2013.

[23] Rodrigo Costas, Tina Nane, and Vincent Larivière. Is the year of first publication a good proxy of scholars' academic age? In *ISSI*, 2015.

[24] Hastie Trevor, Tibshirani Robert, and Friedman JH. The elements of statistical learning: data mining, inference, and prediction, 2009.

[25] Nir Friedman, Moises Goldszmidt, and Abraham Wyner. Data analysis with bayesian networks: A bootstrap approach. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 196–205. Morgan Kaufmann Publishers Inc., 1999.

[26] David S Watkins. Understanding the qr algorithm. *SIAM review*, 24(4):427–440, 1982.

[27] Fred B Bryant and Paul R Yarnold. Principal-components analysis and exploratory and confirmatory factor analysis. 1995.

[28] Carl Edward Rasmussen. The infinite gaussian mixture model. In *Advances in neural information processing systems*, pages 554–560, 2000.

Appendix A

R-code

```



#Hartemink method(see report for explanation)
#args:
#list= all possible values for the number of clusters
#columnname= number of the column you would like to discretize
#dataset= the dataset you want to use for discretization
Hartemink<-function(dataset,breaks,list){
  disc=discretize(dataset[list],method =
                    "hartemink",ibreaks=30,breaks=breaks);
  dataset[list]=disc;
  return(dataset);
}






# k-mean clustering for discretization of the variables and
    determine the
#right number of cluster with the silhouette (See report for
    explanation)
# statistic
#args:
#list= all possible values for the number of clusters
#columnnumbers= number of the column you would like to discretize
#dataset= the dataset you want to use for discretization
kmean<-function(list,columnnumber,dataset){
  max=-10000000;
  k=0;
  set=dist(dataset[columnnumber]);
  for(val in list){
    vector=kmeans(dataset[columnnumber],val,nstart = 20)$cluster;
    sil=silhouette(vector,set);
    if(mean(sil[,3])>max){
      k=val;

      max=mean(sil[,3]);
    }
  }
  return(k);
}

# k-mean clustering for discretization and returns the new
    dataframe
#args:
#list= all possible values for the number of clusters
#columnnnumber= the columnnumbers
#dataset= the dataset you want to use for discretization
kmeandataset<-function(dataset,list,columnnumbers){
  for(i in columnnumbers){
    bestk=kmean(list,i,dataset);
    dataset[i]=as.factor(fitted(kmeans(dataset[i],centers=bestk,
        nstart=20)));
  }
  return(dataset)
}

# Interval-method for discretization (See report for explanation)
#args:
# dataset: The dataset you want to use for discretixation
# breaks: number of cluster
# list: vector of columnnumbers you want to discretize.
#This list will have size 1 most of the time.
IntervalMethod<-function(dataset,breaks,list){
  for(val in list){
    dataset[val]= discretize
    (data.frame(as.numeric(unlist(dataset[val]))),
                              method="interval",breaks=breaks);
  }
  return(dataset);
}

# quantile-method for discretizationreturn(k);
#args:
# dataset: The dataset you want to use for discretixation
# breaks: number of cluster
# list: vector of columnnumbers you want to discretize.
#This list will have size 1 most of the time.
QuantileMethod<-function(dataset,breaks,list){
  for(val in list){
    dataset[val]= discretize(data.frame(as.numeric(unlist(dataset[
        val]))),
                              method="quantile",breaks=breaks);
  }
  return(dataset);
}


# Hillclimbing algorithm with different start positions.
#This is done such that we can prevent that we get a local optimum
    as
# a result
```

```
#args:
# dataset: The dataset you want to use for discretixation
# method: score you want to use
# number: number of times you want to start with different DAGs
HillClimbing<-function(dataset,method,number){
    nodes=names(dataset);
    score=1000;
    for(val in 1:number){
      bn.hc=hc(dataset,start=random.graph(nodes),score=method);
      newscore= score(bn.hc,dataset);
      if(newscore<score){
        score=newscore;
        network=bn.hc;
      }
    }

  return(network);
}



Tabu<-function(dataset,method){
  return(tabu(dataset,score=method))
}

Compare<-function(dataset,method,number){
  hc=HillClimbing(dataset,method,number);
  tabu=Tabu(dataset,method);
  if(score(hc,dataset)<=score(tabu,dataset)){
    return(hc);
  }
  else{
    return(tabu);
  }
}


CrossValidation<-function(dataset,methods,nstart,runs,loss.args){
  args=c();
  for(val in length(methods)){
    k=Compare(dataset,methods[val],nstart);
    plot(k)
    args[val]=bn.cv(dataset,k,runs=runs,loss="pred",
                    loss.args=list(target=loss.args));
  }
  return(args);

}


main<-function(){
  wis=Convert(Canadian_researchers_bn,c(6:16));
  wis=data.frame(IntervalMethod(data.frame(wis),10,c(6:16)));
  wis[1]=data.frame(as.factor(unlist(wis[1])));
  wis[2]=data.frame(as.factor(unlist(wis[2])));
  wis[3]=data.frame(as.factor(unlist(wis[3])));
  wis[4]=data.frame(as.factor(unlist(wis[4])));
  wis[5]=data.frame(as.factor(unlist(wis[5])));
  wis$id_ost=NULL;
  s=CrossValidation(wis,c("k2"),1,1,"refs_paper");
  return(s);
}


Convert<-function(dataset,columns){
  for(val in columns){
    dataset[val]=as.numeric(unlist(dataset[val]));
  }
  return(dataset)
}

make<-function(){
  return(bn.cv(data.frame(Convert(Canadian_researchers_bn[6:16],c
      (1:11))),Compare(data.frame(Convert(Canadian_researchers_bn
      [6:16],c(1:11)))),method =c("bge"),5),list(target="pp_top_
      prop"))
}


Bootstrapping<-function(dataset){
  dataset[5]=NULL;
  for(val in 5:15){
    dataset[val]=data.frame(as.numeric(unlist(dataset[val])));
  }
  args=c();
  replicates=c(100,300,400,1000);
  tresholds=c(0.6,0.8,0.95);
  algorithm=c("tabu","hc");
  k=1;
  for(va in 1:2){
  for(val in 1:length(replicates)){
    for(i in 1:length(tresholds)){
      d=boot.strength(dataset,algorithm = algorithm[va],R=200,m=
          replicates[val],cpdag = TRUE);
      nw=model2network(paste(modelstring(skeleton(averaged.network(
          d[(d$direction >0.5) & (d$direction <0.7),],threshold=
          tresholds[i],names(dataset)))),
              modelstring(averaged.network(d[(d$direction >0.7),],
                  threshold=tresholds[i],names(dataset)))))
      plot(nw)
      args[k]=nw;
      k=k+1;
    }
  }
  }
  return(args);


}

bl=matrix(c("Phd_year","birthyear","P","birthyear","fpy","birthyear
    ",
              "pp_pos_last","birthyear",
            "pp_int_collab","pp_collab","pp_int_collab","fpy","pp_
                pos_last",
            "fpy","pp_pos_last","Phd_year","P",
            "pp_top_prop"),ncol=2,byrow = TRUE,dimnames = list(NULL
                , c("from", "to")))
wl=matrix(c("Phd_year","P","Phd_year","fpy","birthyear","Phd_year")
    ,
          ncol=2,byrow = TRUE,dimnames = list(NULL, c("from", "to")
              ))



GetResults<-function(){
  results=c();
  scores=c();

  first=hc(data.frame(wis),restart=10);
  plot(first);
  results=append(results,(loss(bn.cv(first,data=data.frame(wis),
      loss="mse",loss.args=list(target="pp_top_prop")))));
  scores=append(scores,score(first,data.frame(wis)));
  second=tabu(data.frame(wis));
  plot(second);
  results=append(results,(loss(bn.cv(second,data=data.frame(wis),
      loss="mse",loss.args=list(target="pp_top_prop")))));
  scores=append(scores,score(second,data.frame(wis)))
  third=hc(data.frame(wis),restart=10,blacklist = bl,whitelist = wl
      )
  plot(third);
  results=append(results,(loss(bn.cv(third,data=data.frame(wis),
      loss="mse",loss.args=list(target="pp_top_prop")))));
  scores=append(scores,score(third,data.frame(wis)));
  fourth=tabu(data.frame(wis),blacklist = bl,whitelist = wl)
  plot(fourth);
  results=append(results,(loss(bn.cv(fourth,data=data.frame(wis),
      loss="mse",loss.args=list(target="pp_top_prop")))));
  scores=append(scores,score(fourth,data.frame(wis)));
  d=boot.strength(data.frame(wis),algorithm = "hc",algorithm.args =
       list(restart=10),);
  fifth=averaged.network(d[d$direction >=0.5,],threshold=0.75,names(
      data.frame(wis)));
  plot(fifth);
  results=append(results,(loss(bn.cv(fifth,data=data.frame(wis),
      loss="mse",loss.args=list(target="pp_top_prop")))));
  scores=append(scores,score(fifth,data.frame(wis)));
  d=boot.strength(data.frame(wis),algorithm = "hc",algorithm.args =
       list(blacklist=bl,whitelist=wl,restart=10),,cpdag=FALSE)
  six=averaged.network(d[d$direction >=0.5,],threshold=0.75,names(
      data.frame(wis)));
  plot(six);
  results=append(results,(loss(bn.cv(six,data=data.frame(wis),loss=
      "mse",loss.args=list(target="pp_top_prop")))));
  scores=append(scores,score(six,data.frame(wis)));
  d=boot.strength(data.frame(wis),algorithm = "tabu",);
  seventh=averaged.network(d[d$direction >=0.5,],threshold=0.75,
      names(data.frame(wis)));
  plot(seventh);
  results=append(results,(loss(bn.cv(seventh,data=data.frame(wis),
      loss="mse",loss.args=list(target="pp_top_prop")))));
  scores=append(scores,score(seventh,data.frame(wis)));
  d=boot.strength(data.frame(wis),algorithm = "tabu",algorithm.args
      =list(blacklist=bl,whitelist=wl),);
  eighth=averaged.network(d[d$direction >=0.5,],threshold=0.75,names
      (data.frame(wis)));
  plot(eighth);
  results=append(results,(loss(bn.cv(eighth,data=data.frame(wis),
      loss="mse",loss.args=list(target="pp_top_prop")))));
  scores=append(scores,score(eighth,data.frame(wis)));

  return(data.frame(results,scores));
}

##SELFCOMPOSED ALGORITHM pART1

addtoFrame<-function(ok,k,data){
  r=c();
  for(el in 1:k){
    df=data.frame(matrix(ncol=length(data),nrow=0));
    colnames(df)=names(data);
    for (val in 1:nrow(data)){
      if(ok[val]==el){
        df=rbind(df,data[val,]);
      }
    }
    bn.hc=hc(df,whitelist = wl,blacklist = bl);
    plot(bn.hc)
    r=append(r,modelstring(bn.hc))

  }
  return(r);

}

##Self composed algorithm part 2
checkCluster<-function(o,query,df){
  k=1000;
  s=0;
  for(val in 1:nrow(o)){
    w=EuclideanDistance(o[val,],query);
    if(as.numeric(w)<k){
      k=w;
      s=val;
    }
  }
  return(df[s]);
}


##Self composed algorithm part 3
predictred<-function(test,training,columns,o,df){
  k=c();
  for(val in 1:nrow(test)){
    k[val]=predict(bn.fit(model2network
                            (checkCluster(o,as.numeric(test[val,][
                                columns]),df)),
                            data=training),node="pp_top_prop",data=
                                test[val,])
  }
  print(typeof(k))
  return(MSE(as.numeric(k),test$pp_top_prop));
}

#"Journal score","refs_paper",




TresholdCheck<-fucntion(){
  results=c();
  d=boot.strength(data.frame(wis),algorithm = "tabu",algorithm.args
      =list(blacklist=bl,whitelist=wl),);
  eighth=averaged.network(d[d$direction >=0.5,],threshold=0.5,names(
      data.frame(wis)));
  results=append(results,(loss(bn.cv(eighth,data=data.frame(wis),
      loss="mse",loss.args=list(target="pp_top_prop")))));
  d=boot.strength(data.frame(wis),algorithm = "tabu",algorithm.args
      =list(blacklist=bl,whitelist=wl),);
  eighth=averaged.network(d[d$direction >=0.5,],threshold=0.6,names(
      data.frame(wis)));
  results=append(results,(loss(bn.cv(eighth,data=data.frame(wis),
      loss="mse",loss.args=list(target="pp_top_prop")))));
  d=boot.strength(data.frame(wis),algorithm = "tabu",algorithm.args
      =list(blacklist=bl,whitelist=wl),);
  eighth=averaged.network(d[d$direction >=0.5,],threshold=0.7,names(
      data.frame(wis)));
  results=append(results,(loss(bn.cv(eighth,data=data.frame(wis),
      loss="mse", loss.args=list(target="pp_top_prop")))));
  d=boot.strength(data.frame(wis),algorithm = "tabu",algorithm.args
      =list(blacklist=bl,whitelist=wl),);
  eighth=averaged.network(d[d$direction >=0.5,],threshold=0.8,names(
      data.frame(wis)));
  results=append(results,(loss(bn.cv(eighth,data=data.frame(wis),
      loss="mse",loss.args=list(target="pp_top_prop")))));
  d=boot.strength(data.frame(wis),algorithm = "tabu",algorithm.args
      =list(blacklist=bl,whitelist=wl),);
  eighth=averaged.network(d[d$direction >=0.5,],threshold=0.9,names(
      data.frame(wis)));
  results=append(results,(loss(bn.cv(eighth,data=data.frame(wis),
      loss="mse",loss.args=list(target="pp_top_prop")))));
  plot(c(1:5),results)


}
```

first.R