# Belief Nets for Mental Health

Mathieu Chatelain

MSc by Research in Pattern Analysis and Neural Networks

ASTON UNIVERSITY

September 2005

ASTON UNIVERSITY

# Belief Nets for Mental Health

MATHIEU CHATELAIN

MSc by Research in Pattern Analysis and Neural Networks, 2005

**Thesis Summary**

A Bayesian network is a powerful tool for inference; its structure and parameters can be learnt from data. The objectives of this thesis are first to find the structure associated with the suicide risk of a patient, to compare this structure with that of the expert and then to predict the risk itself. In order to do so, we first learn the structure using algorithms such as **K2** or **MWST**. After having obtained this structure we try to predict the risk of suicide, with this found structure, on testing data. The results, which are quite good, will be discussed in this thesis, but the lack of data for training part has a significant effect.

**Keywords:** Bayesian Networks, Structure Learning, Mental Health

# Acknowledgements

I would like to thank all the people who shared this year with me and helped me through this:

- My supervisor, Ian Nabney for all his advice and all the time he spent with me.
- My second supervisor, Chris Buckingham for all his advice.
- All the staff of the NCRG who welcome us between them.

This year would not have been the same if some people were not here, I speak about:

- All the frenchies:
  - Mathieu Collas alias p0il,
  - Boris Dubuisson alias Letisgo,
  - Etienne Mallard alias EarthquakeProof,
  - Jean Nicolas Turlier alias Ludo.
- Amy Rostron who taught us lots of English and learnt French humour.
- All the PhD students, who were always here to speak and help us.
- Dr Davide Dalimonte who spent time with us and was always enthusiastic.
- Special thanks to my family who have always been by my side even with the distance between us.
- Another special thanks to Eirini Bazaki for the time and talks we shared together.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Mental Health Assessment

In our modern society, stress and pressure have made mental health problems a burning issue. That is why some measures are taken to fight this scourge.

According to [5], the National Health Service (**NHS**), through its New and Emerging Applications of Technology (**NEAT**) programme is interested in providing a tool to help non-specialists, like policemen and social workers, make judgements on the risk (to self and others) of people who present possible mental health symptoms. Such a software tool is called a Decision Support System (**DSS**).

More generally, medical decision support systems are computer programs designed to improve the clinical process of decision making. Intelligent decision support systems need artificial intelligent techniques in order to support professionals of health. The aim of such system is to help people making the best diagnosis and treatment when they don't have access to all information or when it is uncertain. Decision support systems can be active or passive:

- active, when the computer gives advice when it is needed and
- passive, when the professional use it only when they need help.

A decision support system can be judged by its performances (eg. accurate data and reliable solution purposed by the system) but also by its interface. Indeed, with a simple user interface, people won't need a lot of training to use the system.

The Galatean Risk Screening Tool (**GRiST**), introduced in [2], is a **DSS** which asks low level questions. These questions are grouped into 5 different categories and have been chosen by experts to be as specific as possible. However, answering these questions does not require an expert judgement. Thus, the required information can be given accurately by anyone with some understanding of mental health issues. Its aim is to advise which patients present some risk of mental troubles such as suicide, self neglect, harm to self and to others.

**GRiST** is based on a hierarchical tree structure derived from interviews with mental health experts (see further discussion in the next Section). For example, Figure 1.1

extracted from [4], shows the galatea representing knowledge about suicide (see explanation in Sub Section 2.2.1). This tree demonstrates how the risk of suicide can be explained by a combination of different factors. The leaves of the tree are simple questions which can be asked by non experts whereas grey ellipses are summative concepts. For example the node *client's previous attempts* is defined by the questions attached to it: *are there any*, *warnings given* and *triggers*. In a similar way, the representation for self neglect can be found in [5] and [4].

A key point is that we are dealing with health, so misclassification errors must be avoided. Some could be seen as more serious than others. For example, diagnosing a safe patient to be potentially suicidal is less "dangerous" than if a suicidal one is said to be safe; indeed he will be free to commit suicide. But this misclassification should not be disregarded; a safe patient diagnosed to be potentially suicidal will have to go through many interviews and many tests to prove that he is safe, a process that could be really traumatizing.

In [19], the example of cancer detection is discussed: the application has totally imbalanced decision costs. Thus in the construction/training process some errors are more penalized than others, since an error could lead to the death of the patient if a cancer is not diagnosed in time.

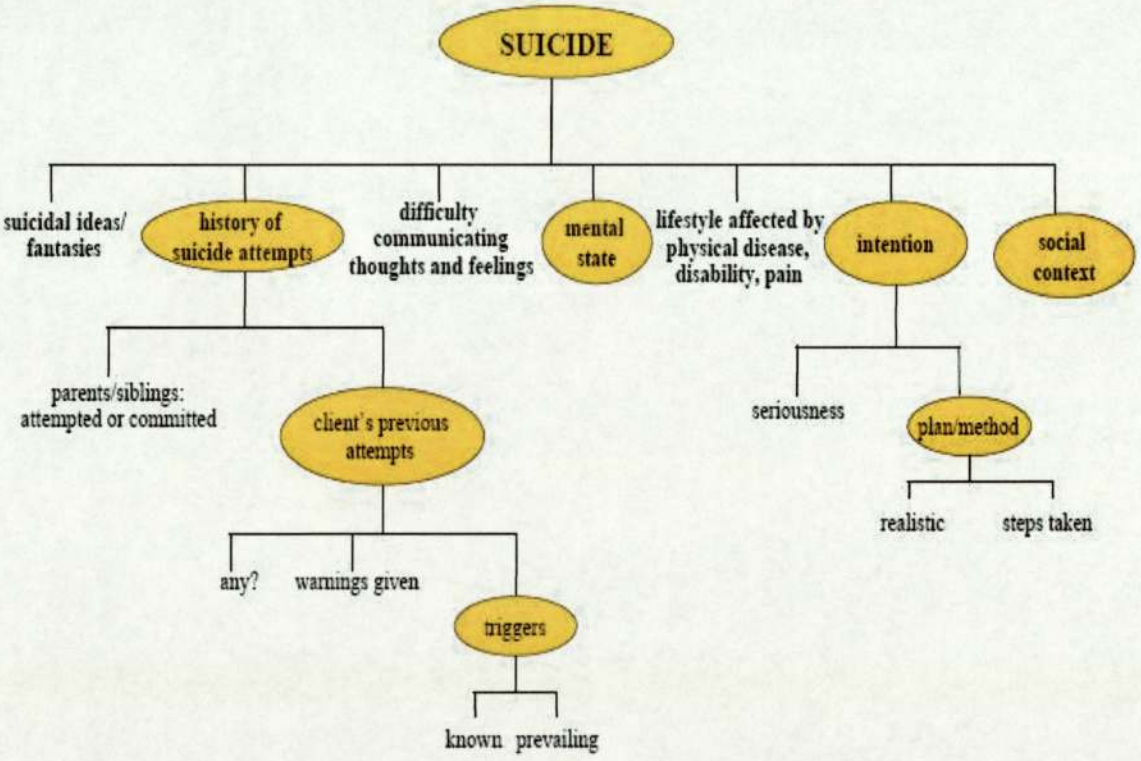



Figure 1.1: Hierarchical galatea structure for suicide.

## 1.2 Initial Problem

As we have seen, the hierarchical tree structure of Figure 1.1 has been realised through meetings and interviews with mental health expert. Soon we shall see, in Section 3.2, that we will represent this knowledge with a *Bayesian Network*. Such a network does not only needs links between variables; it also requires some *probabilities*.

This notion of probabilities is natural for anyone with statistical knowledge but, for an expert in mental health, it is more difficult to grasp the meaning of the probability of an event A given an event B. This point explains why representing the Galatea by a Bayesian network will be difficult and time-consuming a priori. A good discussion of this problem is contained in [19] where it is explained how it is easier to deal with a special scale instead of directly with probabilities. Questions had been reformulated to obtain some conditionally probabilities. Figure 1.2 (extracted from [19] ) is an example of a good approximation of the underlying probabilities and was obtained from the experts. This shows what question and what scale were needed to obtain the probability of an invasion given a tumor with a specific shape and length. For a mathematician, this probability would have been noted $P$(Invasion|Shape, Length) but the explanatory text in this figure is needed for domain experts. Then, when they are asked to provide probabilities, the answers of the experts were much more similar and the strength they associate to their words fits well the current scale.

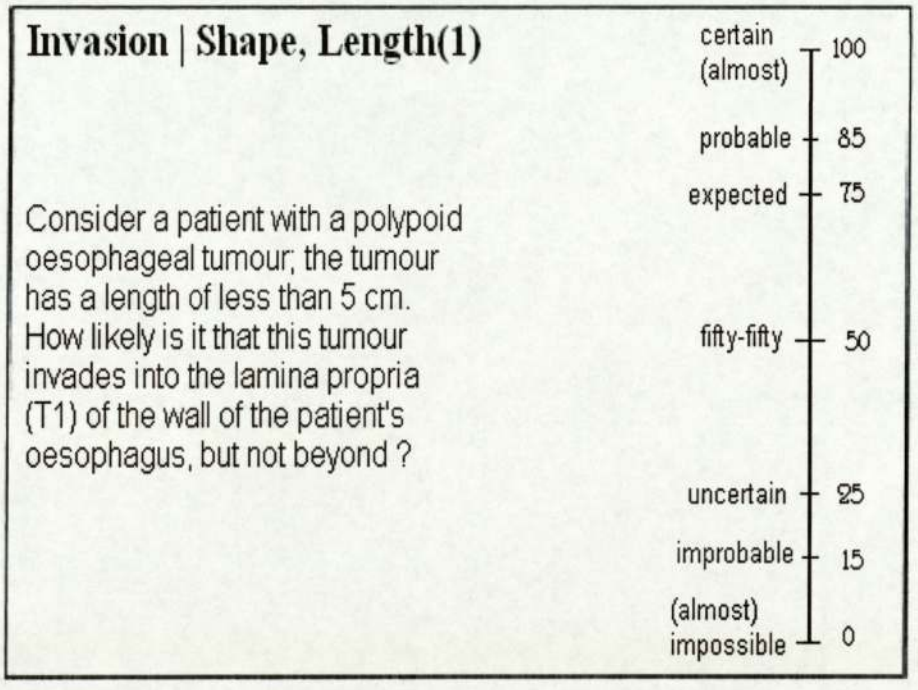



Figure 1.2: Example of the scale for probability elicitation.

This thesis is about mental health risk detection; it aims to predict the risk of suicide of a patient given some information. To do so we will construct a hierarchy among the variables corresponding to different questions. In fact it will be more than a simple hierarchy; it will be a Bayesian network which means that a relationship between two

variables is defined by the presence of a link between them and a certain probability of one given the other. We will learn the structure and the probabilities from the data we have and finally we will try to compare it with the hierarchy of the decision support system developed by hand. By learning the conditional probabilities among the variables we will be able to increase the accuracy of the results; indeed we will try to *classify* some data.

In the second chapter we will introduce the Galatean Screening Tools and we will present what already exists on the subject of mental health detection; then in the third chapter we will explain the mathematical background of the thesis and describe what a Bayesian network is, and what algorithms to train it, exist. The fourth chapter will be devoted to experiments on toy data; we will test the different algorithms on small data sets and finally apply these algorithms on the real data in the last chapter.

# Chapter 2

# Galatean Risk Screening Tools

## 2.1 Project Genesis

Work on this project began in 1999, when a mapping exercise took place to survey the risk screening tools then in use across the Surrey Hampshire Borders NHS Trust. At this time, according to [3], the results showed that some tools were available in a number of services, but few were used every day. Many professionals knew of the existence of tools; nevertheless uncertainties remained over their use and their accuracy was not trusted. Many staff members preferred to use their clinical and professional judgement to interpret the risk; for this reasons, few tools were in use at that time.

According to [4], the Trust was asked to establish a uniform system for risk screening in order to incorporate it into the Care Programme Approach. As many of the patients of the Trust did not show significant risks of suicide or self harm, it was too time-consuming and impractical to subject them all to full detailed tests. Instead the Trust wanted a tool that identified 'high risk' patients for practitioners working in any mental health settings. Such a low-level, intuitive data-gathering tool should not require expert knowledge.

In order to create such a tool, a project was set up with financial support from the NHS Beacon initiative. It officially commenced in November 1999. The outcome was **GRiST**, which can be used by any health care professionals without lengthy training or specialised clinical expertise capacities. The information profile generated by **GRiST** would alert the assessor if needed. In that case, a more detailed risk assessment will be used. **GRiST**provides documented evidence of the decision-making process in risk assessment. It aims to help practitioners to consider the risk factor systematically in order to supplement good practice in patient assessment.

## 2.2 Development and Properties

### 2.2.1 The Galatean Model

Buckingham and Adams have argued in [4] that many clinical decisions can be viewed as classification tasks. The descriptive attributes, called cues, are used to assign patients to one specific category. This approach to clinical decision making is useful as it prevents confusion between practitioners from different backgrounds and training. They can communicate with the same terms rather than losing time and being misunderstood just because of differences of terminology.

The Galatean model is a theory of psychological classification based on the prototype model theory ([4]). In a prototype model, classes are represented by a single, most typical member. This member can be viewed as the class central tendency. For example, if clinicians use the prototype model's representation of suicide, they would take a single member from people having previously attempted suicide. This patient's cues will most likely have occurred amongst most of the known members of the suicide-attempting class; for that reason, the prototype is a hypothetical summary representation of the class. Actually, no one is likely to possess all of these typical cues. The prototype is used for comparison with new clients, to determine how similar they are to it and, therefore, how likely they are to be also a member of the suicide-attempting class. Another way to say it is, how high a risk of attempting suicide they present.

As explained in [4], for prototypes, the most typical class member which is the class representation has, by definition, cues with the highest probability of occurrence within the class, $P(\text{cues}|\text{Class})_{max}$. The prototypical suicide-attempter would be the client whose cues are the most frequent values within its class. Instead of representing the most likely cues amongst all people *known* to be suicide-committers, the Galatean model represents cues of people representing the highest risk of *becoming* a member of this class. This "perfect" object does not maximize the conditional probability of its cues given the class; but maximizes the conditional probability of the class given its cues, $P(\text{Class}|\text{cues})_{max}$. These class representations are called "galateas" in reference to Pygmalion's statue of his perfect woman; their cues are the "perfect" ones, an exception and not the most typical cues.

**GRiST**is a web-based tool so given the answer of the form it will estimates the risk of the patient by comparing with its existing base. This estimation is given in an understandable result and also purpose a treatment that should be followed.

### 2.2.2 Evolution of the Tool

First of all, the hierarchical tree of Figure 1.1 was not easy to construct and has required many interviews. A first prototype of the tree concerning self neglect can be seen in Figure 2.1, extracted from [4].

Finally, after lots of work and interviews, this graph has become much easier to deal with and clearer as well; the final version is shown in Figure 2.2.

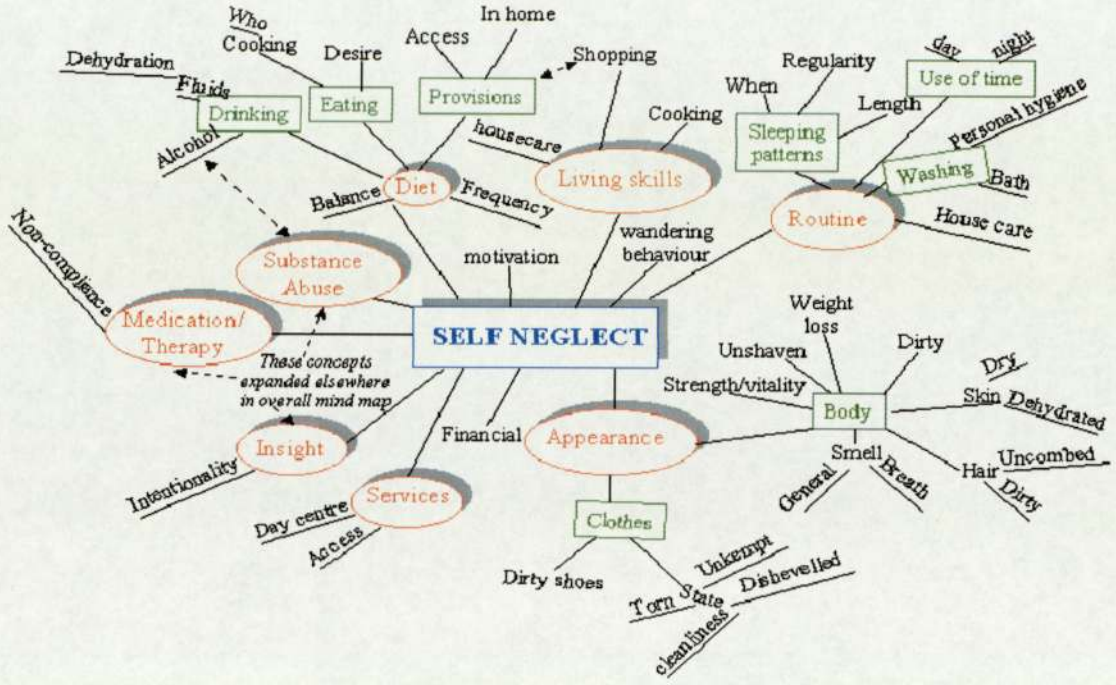



Figure 2.1: Mind map of the self neglect part of the domain.

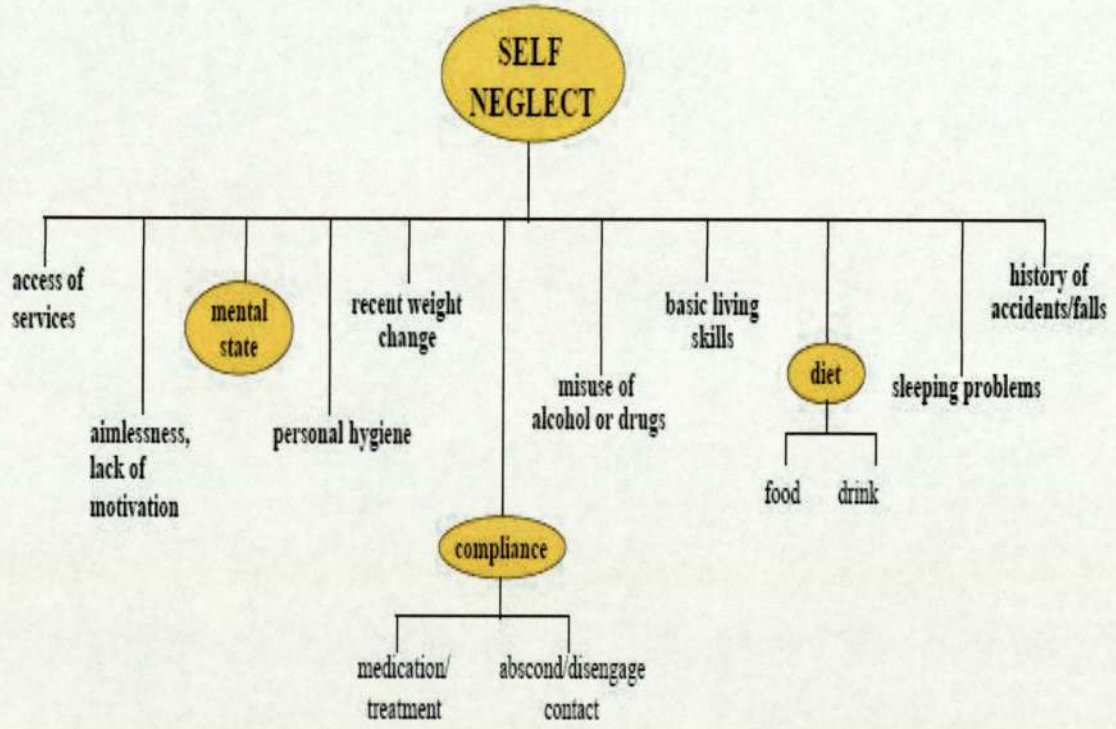



Figure 2.2: Final version of the self neglect part of the Galatea.

### 2.2.3 Properties of the Tool

Another interesting point to notice is that some information is not exact: to quantify the client's suicidal ideas or fantasies there is no objective measure without uncertainty such as, for example, a person's height or weight. Given this observation, we can better understand that **GRiST** does not require 'yes' or 'no' answers. Most of them are numbers between 0 and 10.

- 0 means not at all,
- 10 means certainty.
- a special answer 'don't know'.

An excerpt from the questionnaire can be found in Figure 2.3, extracted from [3]. In [19], there is a discussion on how most experts give the same figure on such a scale and so decrease the uncertainty of probabilities. According to this article, it seems that people feel much more comfortable with a scale than with only a percentage whose meaning is really obscure.

The relationship between answers "0" and "don't know" is really important. Consider, in Figure 2.3, the question "To what extent does the client have a serious intention to commit suicide?". For a given patient, some clinicians may assume that as no references to this question have been expressed, the "zero" extent is justified, whereas other clinicians will judge that not enough information has been collected during the interview, so they will mark "don't know". According to [3], assessors are prepared to make a judgement about the importance of an item if it relates to the actual manifestation of the client. Thus they might reasonably formulate an opinion from the client's behaviour but not if the issues are linked with historical matters. In this way, the assessor will more easily give a zero rating to issues about the present state of the client if there is no evidence to infer them from the interview. They consider "no news" as "good news" and they would rather give a zero rating than taking no information as an indicator of ignorance. This behaviour is the desired one in risk assessments, indeed the "don't know" answers have to be considered as missing information. Too much missing information would seriously jeopardize confidence in the assessments. It also corresponds with common sense; proving the absence of something is not an easy task. The best way to justify this choice is, given a thorough assessment, to consider a failure to detect the cue as a good cause for believing it is not present. Finally, there is a really thin line between non-detection and absence of knowledge; that is why the relation between "don't know" and "0" answers, and how clinicians conceptualise it, needs monitoring.

Another interesting point about **GRiST** also highlighted in [3], is that some of the questions are subsidiary ones, that is, they depend upon parent questions. They should not be answered except if the root question has a particular value. Given this observation, it is normal to have some missing values in the data. For example, in Figure 2.3 it is obvious that the question "If a plan exists, to what extent have steps been taken to implement it?" would not need an answer if the previous question "To what extent does the client have a realistic plan or method in mind?" received a negative answer.

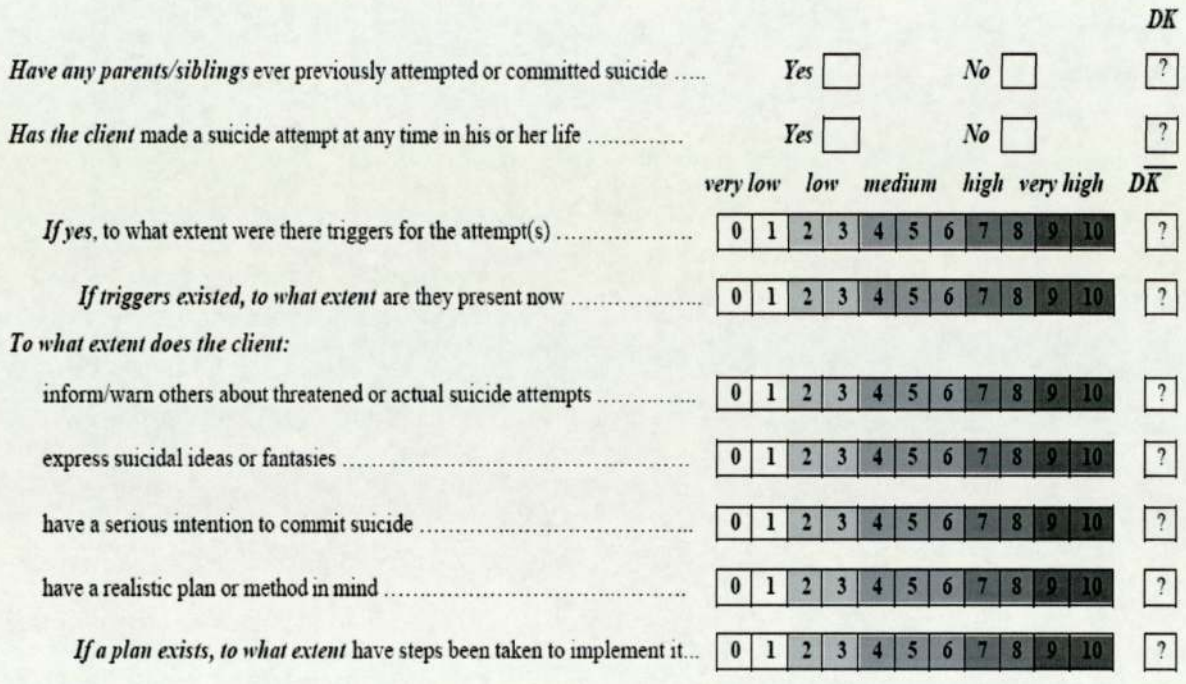



Figure 2.3: Suicide questions in **GRiST**.

In this Chapter we have described a tool developed on the basis of interviews with experts: the Galatean Risk Screening Tool. Its aim is to help predict risk of mental problem, such as suicide, without requiring special training or lot of knowledge in mental health. The thesis objective is, using mathematical theory, to create a software tool that is also capable of predicting risk and then to compare the attribute hierarchy of the software with the expert's one. The following chapter describes the mathematical background used in this thesis.

# Chapter 3

# Structure Learning in Bayesian Networks

## 3.1 Introduction

Over the last few decades, Bayesian belief networks, introduced by Pearl in 1986 [14], have become an important tool for representing and reasoning with uncertainty. Systems based on Bayesian networks have been used in many different areas, from medical diagnosis to computing the prices of shares [17].

Despite these successes, a major obstacle to using Bayesian networks lies in the difficulty of constructing them in complex domains. Indeed it can be a very time-consuming and error-prone task to specify a network for a specific problem.

Some recent research have led to methods for learning the structure of Bayesian networks from data. These new techniques are still evolving but they have been shown to be remarkably effective for some data-analysis problems [8].

Decision trees, artificial neural networks and expert systems are some techniques available for data analysis; many other techniques exist to deal with problem such as density estimation, classification, regression and clustering. So why should we use Bayesian networks and Bayesian inference? At least three answers emerge from [8].

1. For incomplete data sets, Bayesian networks are more usable than other statistical techniques, for which missing values have to be estimated. For example, if we consider a classification problem where two inputs are strongly anti-correlated, if all the inputs can be measured in every case, supervised learning techniques will not face any problem. Nevertheless, when one of the inputs is not observed the results will become dramatically inaccurate. Bayesian networks do not suffer from such a problem.

2. Bayesian networks can deal with causal relationships. This is useful when we try to understand completely a problem domain, for example, during exploratory data analysis. Another interesting point with a causal relationship, is that it allows one to make predictions in the presence of interventions. For example, a student may want to know if it is worthwhile to spend more time working in the

library in order to increase his marks. To answer this question, the student has only to determine whether or not spending time working in the library is a cause for increased marks, and to what degree. The use of Bayesian networks helps to answer such questions. In a Bayesian network, arcs are oriented from cause to effect.

3. Bayesian networks used with Bayesian statistical techniques makes the combination of domain knowledge and data easier. When we perform a real-world analysis, an important thing is the *prior* or domain knowledge, especially when we have only a small amount of data. A proof of the importance of prior knowledge, is that some expert systems can be built from it. As said before, Bayesian networks have a causal semantic that makes causal prior knowledge very easy to encode. Moreover, as we will see further in Section 3.2, Bayesian networks encode the strength of causal relationships with *probabilities.*

## 3.2 Bayesian Networks

All definitions in this chapter are taken from [11].

**Definition 1**

$\mathcal{B} = (\mathcal{G}, \theta)$ *is a Bayesian network if* $\mathcal{G} = (\mathbf{X}, \mathrm{E})$ *is a* **D**irected **A**cyclic **G**raph (**DAG** ) where the set of nodes represents a set of random variables $\mathbf{X} = \{\mathrm{X_1}, \ldots, \mathrm{X_n}\}$, and if $\theta_i = [P(X_i/X_{Pa(X_i)})]$ is the matrix containing the conditional probability of node $i$ given the state of its parents $Pa(X_i)$.

When we refer to parents of $X_i$ we speak about all the nodes $X_j$ such as there is an arrow from $X_j$ to $X_i$.

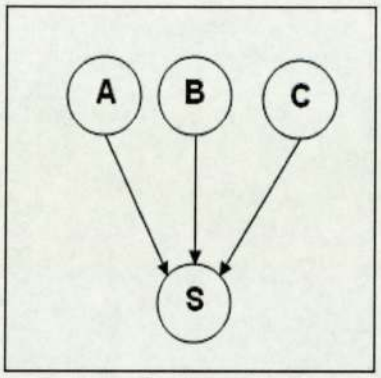
A Bayesian Network for $\mathbf{X} = \{\mathrm{X_1}, ..., \mathrm{X_n}\}$ consists of:

- a network structure, $\mathcal{G}$, that encodes a set of conditional dependencies about variables in $\mathbf{X}$. Each of its nodes represents a variable of $\mathbf{X}$. A lack of arc joining $X_j$ to $X_i$ in $\mathcal{G}$ denotes that these two variables are not directly dependant.
- a set $\theta$ of local **C**onditional **P**robability **D**istributions (**CPD**) associated with each variable.

Together these components define the joint probability distribution for $\mathbf{X}$ , which is given by:

$$P(\mathbf{X}) = \prod_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{P}(\mathbf{X_i}|\mathbf{Pa}(\mathbf{X_i})). \tag{3.1}$$

Figure 3.1 represents a simple Bayesian network with the conditional probability table associated to variable S. Here S is a discrete variable; if it were a continuous variable we would not have a table but a density function. We shall focus on discrete variable in this thesis.




| A | B | C | $Pr(S\|A,B,C)$ |
|---|---|---|---|
| 1 | 1 | 1 | 0.95 |
| 1 | 1 | 0 | 0.95 |
| 1 | 0 | 1 | 0.20 |
| 1 | 0 | 0 | 0.05 |
| 0 | 1 | 1 | 0.00 |
| 0 | 1 | 0 | 0.00 |
| 0 | 0 | 1 | 0.00 |
| 0 | 0 | 0 | 0.00 |

Figure 3.1: A simple network and the associated **CPD** for S.

## 3.3 Structure Learning

### 3.3.1 Introduction

As we have seen, a Bayesian network is composed of a structure and a set of parameters; that means that two different networks could have the same structure but different conditional probability tables. For that reason learning a network from data is a 2 step process:

1. find the structure;
2. find the parameters.

In this thesis we will focus on structure learning rather than on parameter learning; indeed we want to compare the structure learnt from the data with the structure given by the experts.

Basically, there are two main methods to learn structure from data:

- **constraint-based approach**, starting with a fully connected graph, we remove edges if certain conditional independencies are measured in the data.
- **Search-and-score approach** (most popular), we perform a search through the space of possible **DAG**s, and either return the best one found, or return a sample of the models found.

An intuitive idea for learning structure is to consider all the possible **DAG**s, and, given a scoring function, to mark all these **DAG**s before finally choosing the best one. This idea is very good *a priori*, unfortunately, in [16], Robinson in 1977 has shown that the number of possible structures for a Bayesian networks of $n$ nodes is given by the recurrence formula:

$$r(n) = \sum_{i=1}^{n} (-1)^{i+1} C_n^i \;\; 2^{i(n-1)} r(n-i) = n^{2^{O(n)}}. \tag{3.2}$$

The number of possible **DAG**s is super-exponential with respect to the number of nodes. For that reason "brute-force" is useless since it would be much too time consuming and thus computationally infeasible for all but the very smallest networks. To illustrate this, Table 3.1 (extracted from [12]) contains the exact number of possible **DAG** for small networks.

| nodes | **DAG**s |
|---|---|
| 1 | 1 |
| 2 | 3 |
| 3 | 25 |
| 4 | 543 |
| 5 | 29,281 |
| 6 | 3,781,503 |
| 7 | $1.1 \times 10^{9}$ |
| 8 | $7.8 \times 10^{11}$ |
| 9 | $1.2 \times 10^{15}$ |
| 10 | $4.2 \times 10^{18}$ |

Table 3.1: Number of possible **DAG**s as a function of the number of nodes.

In addition to this issue, when we explore the set of possible **DAG**s, to go from one **DAG** to another one we will perform some basic operation; basically we will *add, remove or invert* an arc. This set of operations defines the neighborhood of a **DAG**. As only one dependence changes each time, to reduce the computation required for the evaluation of a neighbor's graph score, it is a good idea to use a sum of local scores.

**Definition 2**

*A score $\mathcal{S}$ is said to be decomposable if it can be written as the sum or the product of functions that depend only on one vertex and its parents. If $n$ is the graph numbers of vertices, a decomposable score $\mathcal{S}$ must be the sum or product of local scores $s$:*

$$\mathcal{S}(\mathcal{G}) = \sum_{i=1}^{n} s(X_i, Pa(X_i)) \quad \textbf{or} \quad \mathcal{S}(\mathcal{G}) = \prod_{i=1}^{n} s(X_i, Pa(X_i)).$$

### 3.3.2 Markov Equivalent Set

As we have seen in Section 3.2, there is a unique joint probability function defined by a Bayesian Networks given by formula 3.1. But the opposite is not true; indeed, using Bayes' rules:

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

We can see that the structures in Figure 3.2 are associated to the same joint probability and all give $A \perp C|B$ (i.e. A is independent of C given B). In other words, given *just* the data, it will be impossible for someone to say which one of these three structures is correct, the one which generated the data. For that reason it is useless to try all of them.

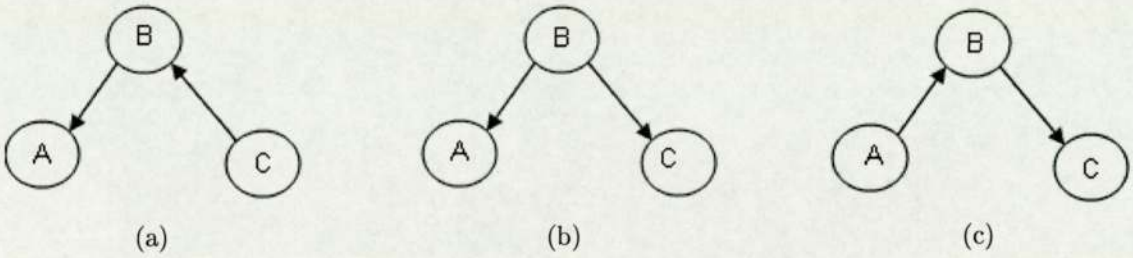



Figure 3.2: Three equivalent structures.

**Definition 3**

*Two* **DAG***s are said to be equivalent (denoted* $\equiv$*) if they imply the same set of conditional dependencies (i.e. they have the same joint distribution). The Markov equivalent classes set (named* $\mathcal{E}$*) is defined as* $\mathcal{E} = \mathcal{A}/\equiv$ *where* $\mathcal{A}$ *is the set of all* **DAG***s.*

Basically, the structures in Figure 3.2 are said to be equivalent since they belong to the same Markov equivalent class. So, given the data we won't be able to choose between these structures which means that the scoring function should give us the same results for all these structures.

**Definition 4**

*A score is said to be equivalent if applied to equivalent* **DAG***s, it gives exactly the same results.*

To sum up, for the moment the space of possible **DAG**s is reduced; indeed why should we consider all equivalent graph given that just from the data we will not be able to differentiate one from the other? This means that we only need to look one instance of each Markov Class.

We are facing a problem here; our goal is to retrieve a structure in order to compare it with the hierarchy given by the expert. From the previous paragraph, we know that from the data we would not be able to differentiate between two equivalent structures,

which means that our task is made more difficult.

The following definition is extracted from [9].

**Definition 5**

*A V-structure in a* **DAG** *is an ordered node triple* $(X_i, X_j, X_k)$ *such that*

- *the* **DAG** *contains arcs* $X_i \rightarrow X_j$ *and* $X_k \rightarrow X_j$, *and*
- *there is no edge between* $X_i$ *and* $X_k$.

The structure, $A \rightarrow B \leftarrow C$ implies $A \not\perp C|B$ which means that if we know the state of node B, the knowledge of node A has a direct impact on node C whereas in the three structures of Figure 3.2 this is not the case. In [11], Verma and Perl showed that two **DAG**s are equivalent *if and only if* they have the same skeleton and the same V-Structures. This allows us to determine in a straightforward way if two networks are equivalent.

**Definition 6**

*An arc is said to be reversible if its reversal leads to a graph which is equivalent to the original. The space of Completed-PDAG s (***CPDAG***s or essential graphs) is defined as the set of Partially Directed Acyclic Graphs (***PDAG***s) that have only undirected or irreversible arcs.*

In a later discussion (Chapter 4) we will not compare directly the graph found and the real one but we will compare their equivalent **PDAG**s and count as an error each misplaced or misdirected arc; basically we will count as a single error any of the following:

- one missing arc,
- one extra arc or
- one misdirected arc.

We have already spoken about scoring functions, it is now time to introduce two classic scores.

## 3.4 Score function

In this section we will use the following notation:

- the Bayesian Network, $\mathcal{B}$ composed of
  - the structure $\mathcal{G}$ and
  - the parameters $\theta$.
- $\mathcal{D}$ is the dataset containing $N$ examples,
- $\theta^{ML}$ is the maximum likelihood parameters,
- $\mathbf{Dim}(\mathcal{B})$ is the network dimension defined in equation 3.7,
- $r_i$ is the size of $X_i$ and
- $pa_i$ is a specific value of the parents$X_i$.

### 3.4.1 Bayes' Score

Introduced in [11], Bayes' score is an equivalent and decomposable score. It is exactly the logarithm of the likelihood of the data. Bayes' score is given by:

$$BAY(\mathcal{B}, \mathcal{D}) = \log P(\mathcal{D}|\mathcal{B}). \tag{3.3}$$

In our case, as we will be looking for the structure and the parameters this score could be written as in formula 3.4 where $\theta^{ML}$ represents the maximum likelihood parameters.

$$BAY(\mathcal{B}, \mathcal{D}) = \log P(\mathcal{D}|\mathcal{G}, \theta^{ML}). \tag{3.4}$$

### 3.4.2 BIC Score

Introduced in [11] as well, the **B**ayesian **I**nformation **C**riterion is also decomposable and equivalent. It comes from principles stated by Schwartz in 1978 and can be expressed by the formula:

$$BIC(\mathcal{B}, \mathcal{D}) = \log P(\mathcal{D}|\mathcal{B}, \theta^{ML}) - \frac{1}{2} \mathbf{Dim}(\mathcal{B}) \log N. \tag{3.5}$$

As we need $r_i - 1$ parameters to describe the conditional probability distribution $P(X_i|Pa(X_i) = pa_i)$,we need $\mathbf{Dim}(X_i, \mathcal{B})$ parameters to describe $P(X_i|Pa(X_i))$ with:

$$\mathbf{Dim}(X_i, \mathcal{B}) = (r_i - 1)q_i \text{ where } q_i = \prod_{X_j \in Pa(X_i)} r_j. \tag{3.6}$$

The Bayesian network dimension $\mathbf{Dim}(\mathcal{B})$ is defined by:

$$\mathbf{Dim}(\mathcal{B}) = \sum_{i=1}^{n} \mathbf{Dim}(X_i, \mathcal{B}). \tag{3.7}$$

The **BIC** score is basically the sum of the likelihood term and a penalty term which penalizes complex networks. As two equivalent graphs have the same likelihood and the same complexity, their **BIC** scores are equivalent.

Using these two basic scores let us now consider some algorithm that can be used to learn the structure of the network.

## 3.5 Structure-Learning Algorithms

### 3.5.1 K2

This algorithm is described in many papers (for example [11] or [12]). Developed by Cooper and Herskovits in 1992 ([7]), the **K2** algorithm is a greedy search algorithm that works as follows.

1. Initially each node has no parents.
2. Order all the nodes and start step 3 with node 1.
3. Add incrementally those parents whose addition most increases the score of the resulting structure.
4. When the addition of a single parent does not increase the score, stop adding parents to the node.
5. Go to step 3 with the next node in the order until all the nodes have been treated.

The fixed order needed by this algorithm, avoids some arcs; indeed if the order is $1-2-3-4$ then node 1 can not have any parents whereas node 4 could have nodes 1, 2, and 3 as parents.

This algorithm belongs to the class of *search and score algorithm* as defined in subsection 3.3.1.

The basic idea of the **K2** algorithm is to maximize the structure probability given the data. In order to do so, we can use the fact that:

$$\frac{P(\mathcal{G}_1|\mathcal{D})}{P(\mathcal{G}_2|\mathcal{D})} = \frac{\frac{P(\mathcal{G}_1,\mathcal{D})}{P(\mathcal{D})}}{\frac{P(\mathcal{G}_2,\mathcal{D})}{P(\mathcal{D})}} = \frac{P(\mathcal{G}_1,\mathcal{D})}{P(\mathcal{G}_2,\mathcal{D})}$$

and the following result given by Cooper and Hersovits in 1992 :

**Theorem** : *If $N_{ijk}$ is the number of data points where $X_i$ has the value $x_{ik}$ and $Pa(X_i)$ is instantiated in $pa_{ij}$ and $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$ then*

$$P(\mathcal{G}, \mathcal{D}) = P(\mathcal{G})P(\mathcal{D}|\mathcal{G}) \text{ with } P(\mathcal{D}|\mathcal{G}) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \tag{3.8}$$

where $P(\mathcal{G})$ is the prior probability of the structure $\mathcal{G}$.

Equation 3.8 can be viewed as a measure of the quality of the network given the data and is named the *Bayesian measure.*

Given a uniform prior on structures, the quality of a node $X$ and its parents can be evaluated by the local score:

$$s(X_i, Pa(X_i)) = \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \tag{3.9}$$

Cooper and Hersovits have also shown that we can reduce the size of the search space using a node order. According to this order, a node can be a parent only of a node which is after it in this order. The search space becomes the subspace of all the **DAG**s admitting this order as topological order.

The **K2** algorithm tests parent insertion according to this order described previously. The first node cannot have any parent, and for other nodes, we choose the parent set that best upgrades the score; that means we add only the parents which most increase the score.

Heckerman et al. have proved in [9] that the Bayesian measure need not be *equivalent* and has proposed the BDe score, which is basically the Bayesian measure with a specific prior on parameters, in order to avoid this. It is also possible to use the **BIC** score in the **K2** algorithm.

### 3.5.2 PC

This algorithm, also described in [12] and [11] uses a different approach from the **K2** algorithm. Indeed the idea is not to find dependencies but to find *independencies.*

This algorithm belongs to the class of *constraint based algorithm* that were defined in sub-section 3.3.1.

The PC algorithm, which was first developed by Spirtes, Glymour, and Scheines in 1993 [18], is a faster version of the IC algorithm introduced by Pearl and Verma in 1991 [15]. Basically, it computes many conditional independence tests, and combines these constraints into a **PDAG** to represent the whole Markov equivalence class.

This algorithm works as follow:

- First, start with a full connected graph.
- Using a statistical test evaluate whether or not there is a conditional independency between two variables.
- If independency has been found remove the corresponding arc.

This algorithm was not used in this research because its results are not accurate; it is usually to give someone information of independencies among variables rather than to find a structure.

An interesting point in Figure 1.1 is that the structure given by the expert for suicide, which is precisely the one the project is focused to retrieve from the data, is a *tree*. The following definition is extracted from [1].

**Definition 7**

*A connected undirected graph is a tree if for every pair of nodes there exists a unique path.*

As a tree is a particular case of a **DAG** the previous algorithms could be used to retrieve it; the next algorithm is not a **DAG** learning algorithm but a *tree* learning algorithm : The **M**aximum **W**eight **S**panning **T**ree.

### 3.5.3 MWST

This algorithm is not recent and one of its variations has been proposed by Chow and Liu in 1968. This variation, which has the name of its creators, is fully described in [6] and also in [11]. This method associates a weight to each edge. The weight could be either the mutual information between the two variables or the score variation when one node becomes a parent of the other (see Heckerman et al. in 1994 [9]) or even the **BIC** score. When the weight matrix is created, the standard **MWST** algorithm gives an undirected tree that can be oriented with the choice of a root.

The main idea of this algorithm is to approximate optimally an $n$-dimensional discrete probability distribution by a product of second-order distributions. In this sub-section we will denote by $P(x)$ the true distribution and by $P_t(x)$ the approximation by the tree. Our goal is to make $P_t(x)$ as close as possible to $P(x)$ and it is precisely from the mutual information that we will characterize this closeness.

Definitions in this section are taken from [6].

**Definition 8**

*The mutual information $I(x_i, x_i)$ between two variables $x_i$ and $x_j$ is given by:*

$$I(x_i, x_j) = \sum_{x_i, x_j} P(x_i, x_j) \log \left( \frac{P(x_i, x_j)}{P(x_i)P(x_j)} \right). \tag{3.10}$$

It is well known that $I(x_i, x_j)$ is non-negative. In the graphical representation of dependence relations, to every branch of the dependence tree we assign a branch weight $I(x_i, x_{j(i)})$. Given a dependence tree $t$, the sum of all branch weights is a useful quantity for determining the closeness of two trees.

**Definition 9**

*A maximum-weight dependence tree is a dependence tree t such that for all $t'$ in the set of all possible dependence tree,*

$$\sum_{i=1}^{n} I(x_i, x_{j(i)}) \geq \sum_{i=1}^{n} I(x_i, x_{j'(i)}).$$

Our goal can now be stated as follows. A tree probability distribution $P_t(x)$ is an optimum approximation to $P(x)$ *if and only if* its dependence tree $t$ has maximum weight

Given samples $(x_1, x_2 \ \ldots \ x_N)$,

1. Start with an empty structure.
2. Compute the mutual information matrix $I_{ij} = I(x_i, x_j)$.
3. Let k defined by the pair $(i_k, j_k)$ such that $I_{i_k j_k} = \max \ I_{ij}$.
4. Put $I_{i_k j_k} = 0$.
5. If the vertex $i_k$ or $j_k$ is not in the structure yet and if the addition of the arc $j_k \rightarrow i_k$ does not lead to a cycle then add it.
6. Go back to step 3 until $I_{ij} = 0, \ \forall (i, j)$ or we have already $N - 1$ arc.

To compute correctly the mutual information you should know entirely the probability distribution for each pair of variable (e.g. to know $P(x_1, x_2)$ for all value of $x_1$ and $x_2$). In our case we won't have such knowledge so we will estimate the distribution from the data statistics by computing the frequency of all cases. Basically we will count the number of occurrences of each case and divide this number by the total of numbers observations. Just in order to avoid any problem and division by zero in the

logarithm of formula 3.10, we can put a Dirichlet prior in order to smooth these frequencies. Basically what we will do is to add 1 occurrences for all the possible values, in other word if there were 3 binary nodes (A,B and C) that mean $2^3$ possibilities and if we have really observe only 2 cases (first case: $\{A = 1, B = 1, C = 0\}$, second case: $\{A = 1, B = 0, C = 0\}$), then we will 'fake' to have observed $2 + 2^3$ cases the 2 real cases and the $2^3$ virtual just to make sure that there won't be any $\log\left(\frac{P(x_i,x_j)}{P(x_i)P(x_j)}\right)$ term which will corrupt our computation (in our case P(C=1) without the Dirichlet prior would have been 0).

In this chapter we presented a mathematical way to represent data and the links between them: a Bayesian network. We have then seen some algorithm to learn the structure of such a network directly from the data. We will now use this Bayesian network to retrieve an unknown structure and compare it with the Galatean model introduced in the first chapter.

# Chapter 4

# Experiments in Structure Learning on Toy Data

In this chapter we will describe some experiments on toy data, and the next chapter will deal with real data. Basically all the experiments are similar: from a network (which is known in this chapter) are generated some data. Given this data, the aim of the experiment is to compare the structure of networks found with **K2** algorithm and the **MWST**. All the scores introduced on Chapter3.2 will be used in order to test if one of them gives better results than the others:

- Bayes' score,
- **BIC** score and
- mutual information.

In all the experiments, we have used the useful **B**ayesian **N**etwork **T**oolbox (**BNT**) written by Kevin Murphy and also very well documented in [12]. This toolbox needs the Matlab program and also the Netlab toolbox.

## 4.1 Description

We carried out experiments on three different networks.

- The first one, is very small: only 4 nodes and 4 arcs.
- The second network is a 10-node network with many more arcs.
- The third network is a 10-node tree.

The first network, in Figure 4.1, has often be studied; it is the "sprinkler network". We can see that the structure is not a tree so the first thing we have to expect is that

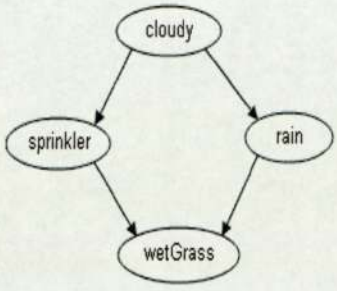



Figure 4.1: First network.

the **MWST** will not give the correct network, but it will be interesting to see whether or not the found arcs are correct. Another point to notice is that given there are only 4 nodes, so there are 543 possible structures (according to Table 3.1) and given that all nodes are binary, the amount of data needed to learn the structure should be relatively limited.

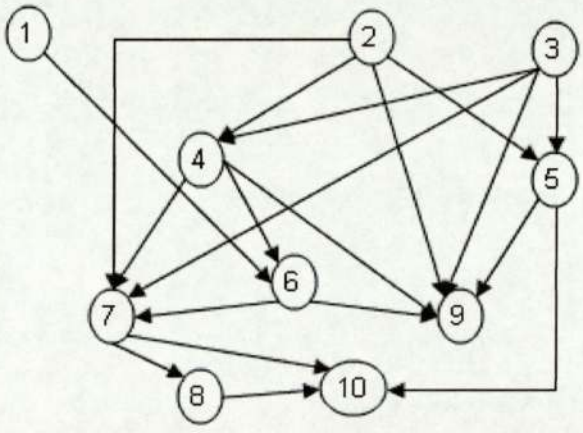



Figure 4.2: Second network.

This second network, in Figure 4.2, is also not a tree, so again the **MWST** would not give a correct network but again we aim to test if the found arc are correct. This network is far more complicated than the previous one. For that reason, we can expect that to retrieve the correct structure we will need a lot more data than in the first case.

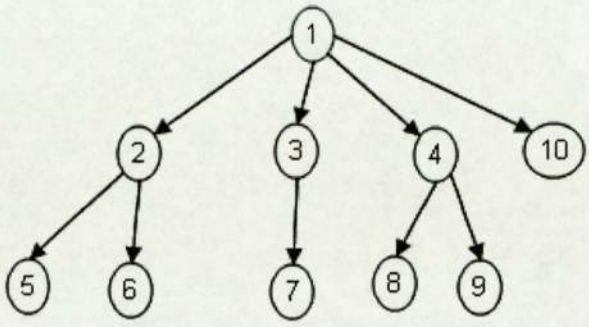



Figure 4.3: Third network.

Finally, the third network is a tree (see Figure 4.3) so we expect good results with the **MWST** algorithm. There are 10 nodes and 9 edges so it is quite similar to the structure given by the experts for suicide in Figure 1.1. Thus the results on this structure will be very useful for the experiment on the true data.

In order to generate the data, as we have fixed the structure and also the conditional probability tables we can create a dataset for the network. Basically, that means that we will generate a possible value for each node of the network. In order to determinate how much training data is needed, we performed many experiments with different sizes of data set look for the size beyond which the results are constant (i.e. one algorithm gives always the correct structure).

## 4.2 Results

In this section we will present tables of the errors using different algorithms and different scores. To obtain this error, we compute the **PDAG** associated to the found network, and count the number of misplaced or misdirected arcs compared to the **PDAG** derived from the true network. The algorithm used are:

- **K2** algorithm with good and bad node orders and with **BIC** and Bayes' score.
- **MWST** algorithm with **BIC** score and mutual information.

The good or bad order for the **K2** algorithm can be defined because, as we generate the data from the networks, we know how the nodes were ordered. If we give the **K2** algorithm this order we will say it was the *good* order; whereas if we give the reverse order we will speak about the *bad* order.

To add some statistical reliability to these results, multiple seeds have been used and the computation time is also calculated. We perform each experiment with more than 100 seeds but we just present here results on 4 seeds chosen because they give the

best answer (i.e. the resulting structure are found correctly with less data). Indeed the choice of the seed is important because every time a choice is made "randomly" it is in fact defined by the seed:

- first of all when we generate the data from the network the choice to generate it is random.
- In the **K2** algorithm if the addition of several parents lead to the same maximal increase of the score the parent is choose randomly.
- Finally in the **MWST** algorithm if several mutual information has the maximal value one is chosen randomly.

### Sprinkler Network

Consider the found structure and a table containing the number of error (in term of arc misplaced or misdirected) with **K2** algorithm with the correct order.

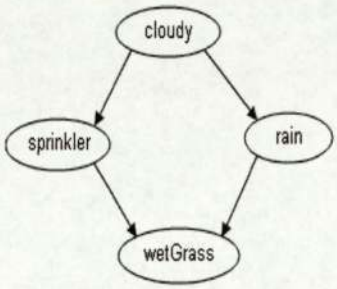



Figure 4.4: Learned network for the **K2** algorithm in good node order for the sprinkler network.

| Seed | 14 | 49 | 51 | 66 |
|---|---|---|---|---|
| BAYE | 0 | 0 | 0 | 0 |
| BIC | 0 | 0 | 0 | 0 |
| Time (s) | 0.1402 | 0.1402 | 0.1402 | 0.1402 |

Table 4.1: Incorrect arcs for **K2** algorithm in good node order for the sprinkler network.

The first thing to notice is that the found network structure is perfect. We have used only 50 data point and we can see that for all seeds, the resulting network is accurate and also found in a very short time.

The next experiment will highlight the importance of the node order in the **K2** algorithm; in the following examples the order is exactly the reverse of the correct order.

| Seed | 14 | 49 | 51 | 66 |
|---|---|---|---|---|
| BAYE | 5 | 4 | 4 | 4 |
| BIC | 5 | 4 | 4 | 4 |
| Time (s) | 0.0701 | 0.0701 | 0.0701 | 0.0701 |

Table 4.2: Incorrect arc for the **K2** algorithm in bad node order for the sprinkler network.

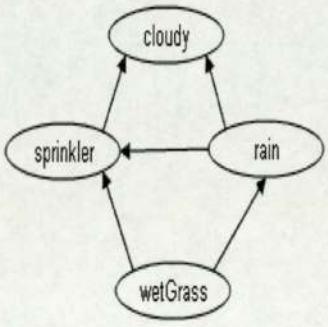



Figure 4.5: Learned network for **K2** algorithm in bad node order for the sprinkler network.

As expected, the network is totally wrong: there is one extra arc and all the others are misdirected. If we train the network with much more data the resulting network is still the same because of the node order.

Another interesting point is the invariance of the resulting network from the score used in the learning; indeed we find the same network if we used either **BIC** score or Bayes' score. That point is quite easy to understand: the main difference between these scores is the penalty term which avoids too complex networks and in this case the network is really simple so this penalty term is small.

Finally, when we used the **MWST** algorithm the resulting structure and error were as follows:

| Seed | 14 | 49 | 51 | 66 |
|---|---|---|---|---|
| Mutual | 2 | 2 | 2 | 2 |
| BIC | 2 | 2 | 2 | 2 |
| Time (s) | 0.3405 | 0.3405 | 0.3405 | 0.3405 |

Table 4.3: Incorrect arc for the **MWST** algorithm for the sprinkler network.

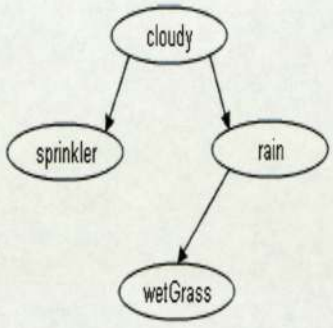



Figure 4.6: Learned network for the **MWST** algorithm for the sprinkler network.

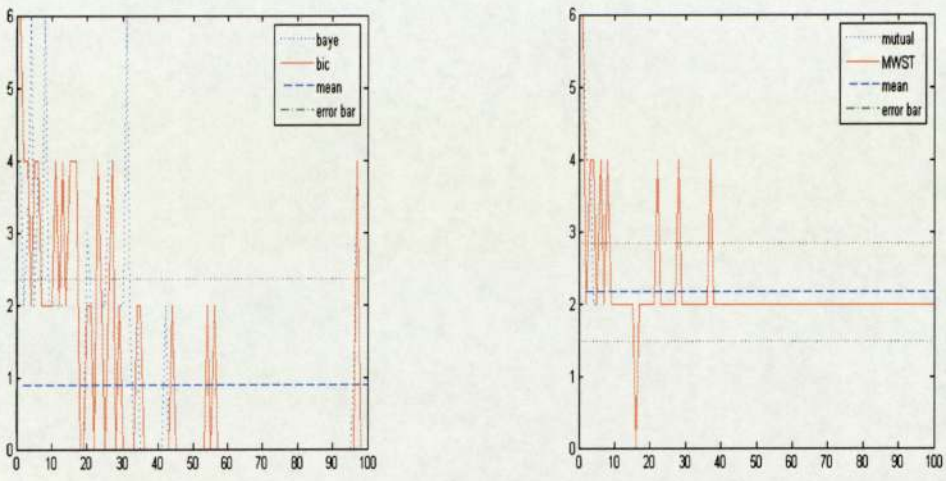
The result should not be a surprise; indeed the original network is not a tree, since, ignoring the arc orientation there is a cycle that could not be present in a tree. Nevertheless we notice that all the arcs found are correct and whatever the number of data points used to perform the experiment it is always this final arc between "sprinkler" and "wet grass" which is missing. That means that the score associated to this arc must be the smallest, either in mutual information or in **BIC** score; so it would be chosen after the others but as the number of arcs is limited in **MWST**, the algorithm stops before the addition of this arc.

Another point to notice is that the time for the experiment is greater than with the **K2** algorithm; this observation confirms the results of [6] that for small networks this algorithm is slower than others.

Note that the number of incorrect arcs is 2 simply because in the **BNT** toolbox, when an arc between node A to node B is reversible, in the associated **CPDAG** there is an extra arc between node B to node A. All arcs found with the **MWST** algorithm are reversible whereas in the true structure there are only 2 reversible arcs.

Finally, Figure 4.7 represents the number of errors as a function of the training dataset size.

On this graph, we can clearly see that with more than 80 data points, the **K2** algorithm gives a constant answer which is the exact network; whereas the **MWST** algorithm gives its final answer after only 10 training data. Nevertheless, except between 20 and 30 training data the results of the **K2** algorithm are better than the **MWST**. The average error for the **K2** is 0.88 with a standard deviation of 1.453 whereas for the **MWST** algorithm the mean error is 2.16 and the standard deviation is 0.24. This figure are the same for both scores.




(a) **K2** (b) **MWST**

Figure 4.7: Number of incorrect arcs in function of the number of training data.

### Second Network

For this experiment, we have used 8100 data points (this number has been found as the smallest data points needed to have a correct answer with the **K2** algorithm) to train the network. Consider the found structure and the error results with the **K2** algorithm with the correct node order.

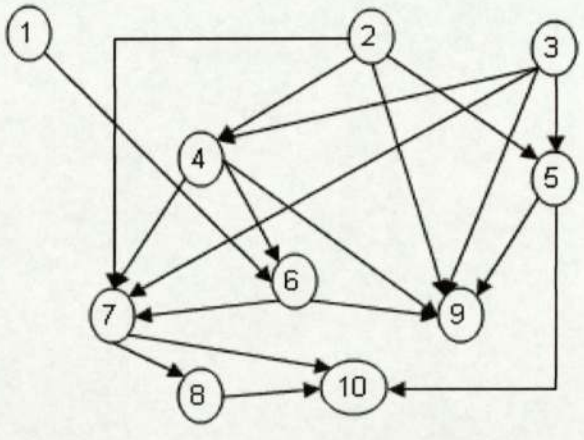



Figure 4.8: Learned network for the **K2** algorithm in good node order for the second network.

The graph in Figure 4.8 has been obtained with seed 14 and is perfect. Table 4.4

| Seed | 14 | 49 | 51 | 66 |
|---|---|---|---|---|
| BAYE | 0 | 5 | 0 | 5 |
| BIC | 0 | 5 | 0 | 5 |
| Time (s) | 7.2248 | 6.9243 | 7.2448 | 7.2348 |

Table 4.4: Incorrect arcs for the **K2** algorithm in good node order for the bigger network.

shows that for different seeds we can have different networks but all are quite close to the true-one. Once more, there is no difference between the results achieved with the 2 scores. Last but not least, we can notice that this experiment is more time consuming than the previous one and we have to keep in mind that the nodes are only binary.

Using the same algorithm but with the reverse node order gives totally different results both for the found structure and for the error.

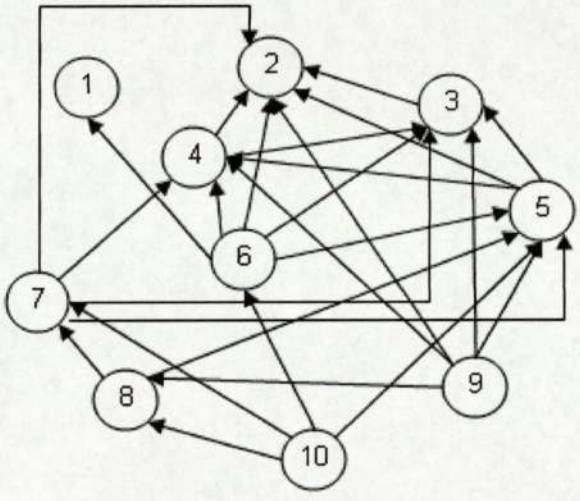



Figure 4.9: Learned network for the **K2** algorithm in bad node order for the bigger network.

| Seed | 14 | 49 | 51 | 66 |
|---|---|---|---|---|
| BAYE | 43 | 41 | 41 | 43 |
| BIC | 43 | 42 | 52 | 43 |
| Time (s) | 7.7956 | 7.4551 | 7.8156 | 7.3950 |

Table 4.5: Incorrect arcs for the **K2** algorithm in bad node order for the bigger network.

In this experiment, we can see that the importance of the node order in **K2** algorithm is even more highlighted than before; the resulting network is far from the true one and we can also see that the scoring function sometimes makes a difference. For example, for seed 49 we can see that the **BIC** score is worse than the Bayes' score: as it penalizes complex networks it avoids some arcs which could be reversible and so not count as error at the end. The time of the experiment is roughly the same as with good node order.

Finally, when the **MWST** algorithm is used to discover this network, the results are as shown in Figure 4.10.

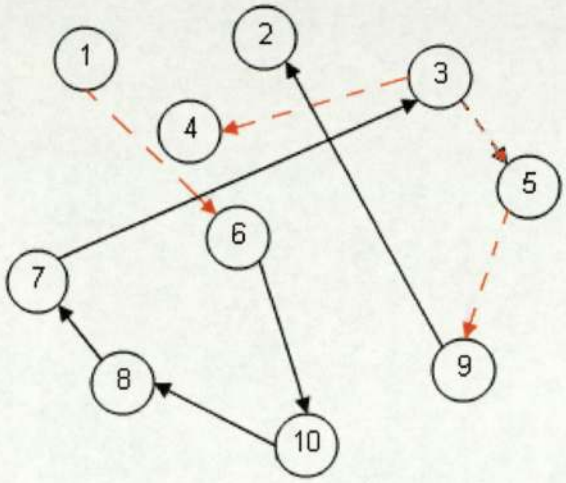



Figure 4.10: Learned network for the **MWST** algorithm for the bigger network.

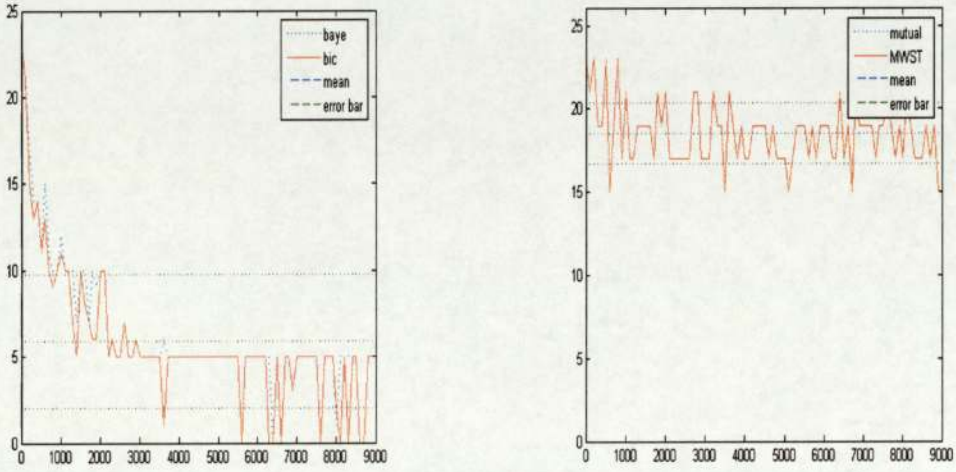
| Seed | 14 | 49 | 51 | 66 |
|---|---|---|---|---|
| Mutual | 21 | 17 | 21 | 17 |
| BIC | 21 | 17 | 21 | 17 |
| Time (s) | 4.5209 | 4.0402 | 4.3807 | 4.2605 |

Table 4.6: Incorrect arcs for the **MWST** algorithm for the bigger network.

As explained in subsection 3.5.3, this algorithm can only find $N - 1$ arcs given there are $N$ nodes. That is why the bad results for this network are predictable there were too many arcs as the true network was not a tree. Arcs in dashed lines are well directed; the other are misdirected and there is also one arc from 6 to 10 that does not exist. There is no difference if we used mutual information or **BIC** score.

According to [11], a good idea when we do not know the order of the nodes is first to apply this algorithm and given the dependency, order the nodes. We can see that in this case the resulting order is really far from the reality, and this is likely to be true for many complex network.

Finally Figure 4.11 represents the number of errors as a function of the training dataset size.




(a) **K2** (b) **MWST**

Figure 4.11: Number of incorrect arcs in function of the number of training data.

We can see on this graph that if we want to be sure to have a correct result using the **K2** algorithm we should use more than 8000 training data points; however, with the Chow Liu algorithm the result are pretty constant, but really bad, whatever the size of the dataset used in the training process. The mean of error for the **K2** algorithm is 6.11 but it's standard deviation is really large 3.9 whereas for the **MWST** it is the opposite, the mean is large(18.4) but the error bars are really small(1.8).

### Tree Network

The experiments done so far are not in favour of the **MWST** algorithm but with the last experiment we will see the practical aspect of this algorithm. For this experiment, we have used 1300 data points to train the network. The resulting error and the found structure with the **K2** algorithm with the correct order for the tree network are shown in Figure 4.3.

| Seed | 14 | 49 | 51 | 66 |
|---|---|---|---|---|
| BAYE | 0 | 0 | 0 | 0 |
| BIC | 0 | 0 | 0 | 0 |
| Time (s) | 4.3907 | 5.0116 | 4.2505 | 4.3406 |

Table 4.7: Incorrect arcs for the **K2** algorithm in good node order for the tree network.

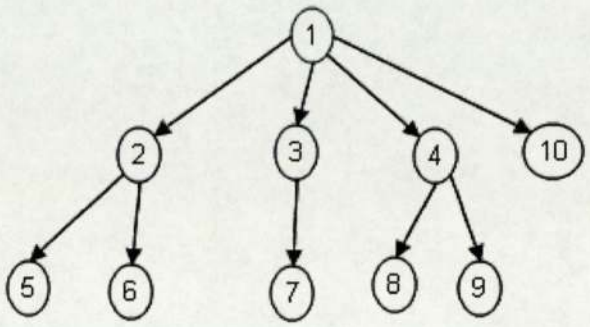



Figure 4.12: Learned network for the **K2** algorithm in good node order for the tree network.

Once again, if we know the correct order of nodes, the **K2** algorithm is accurate but quite time consuming. Whatever the seed and the scoring function used the results are almost constant:

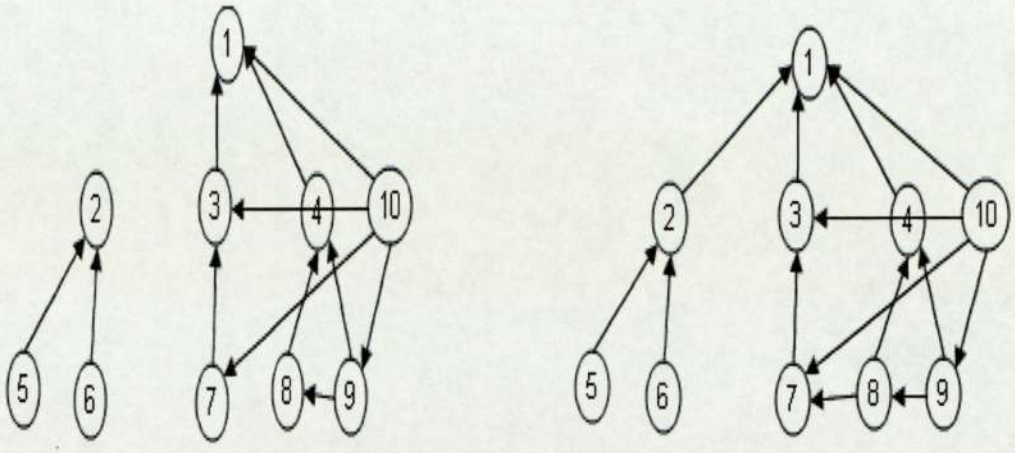
- every time the network is accurately found and
- the experiment lasts about 4.5 seconds.

Then if we used the same algorithm but with the reverse order we found two different structures depending on the score used.

| Seed | 14 | 49 | 51 | 66 |
|---|---|---|---|---|
| BAYE | 13 | 13 | 11 | 11 |
| BIC | 10 | 13 | 11 | 10 |
| Time (s) | 5.2519 | 5.2820 | 5.4222 | 5.3020 |

Table 4.8: Incorrect arcs for the **K2** algorithm in bad node order for the tree network.

As usual, with a wrong node order, the **K2** algorithm loses all its accuracy. All arcs present are misdirected or should not exist at all. For the first time, we can see a difference between the resulting structure if we used the Bayes' score (Figure 4.13 a) or the **BIC** score (Figure 4.13 b). Basically there are two extra arc with the **BIC** score:




(a) Bayes' (b) **BIC**

Figure 4.13: Learned network for the **K2** algorithm in bad node order for the tree network.

between node 2 and node 1 and between node 8 and node 7 .

Finally, when we used the **MWST** to retrieve this tree here are the resulting error and the structure we obtain:

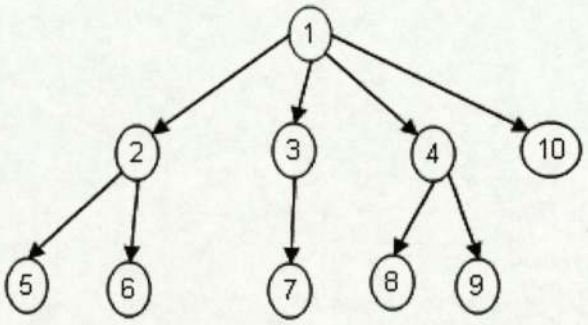


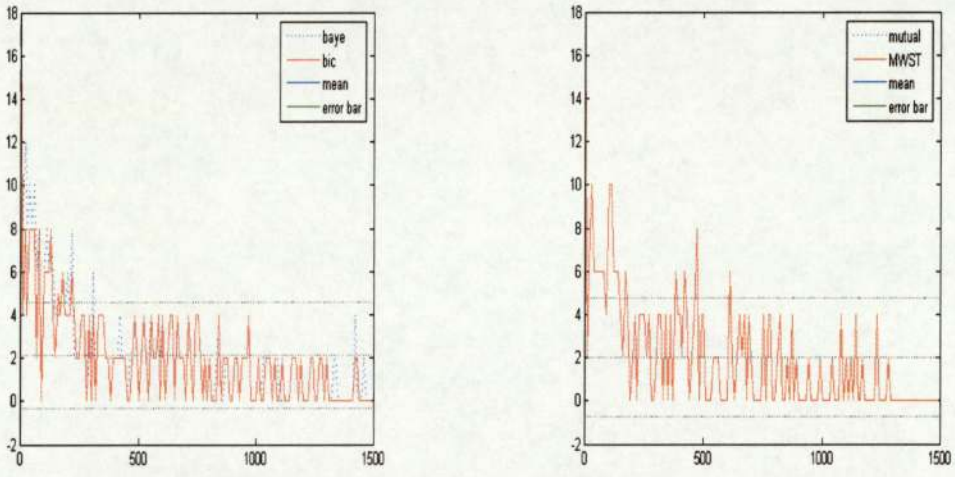

Figure 4.14: Learned network for the **MWST** algorithm for the tree network.

| Seed | 14 | 49 | 51 | 66 |
|---|---|---|---|---|
| Mutual | 0 | 0 | 0 | 0 |
| BIC | 0 | 0 | 0 | 0 |
| Time (s) | 3.1189 | 3.3091 | 3.1890 | 3.0588 |

Table 4.9: Incorrect arcs for the **MWST** algorithm for the tree network.

The resulting network is perfect and found a little faster than with **K2** algorithm with good node order. Again the seed does not modify the results and the scoring function used does not seem to interfere either. This result is encouraging for work on the real data set because the true model should be a tree according to the experts. Nevertheless, the negative point of this experiment is the amount of data used: 1300 points, which is much more than the dataset we have for the suicide case. Indeed we have only 308 data points and in addition, not all the nodes are binary but will have 12 possible values: the scale answer from 0 to 10 plus the "don't know value".

As for the previous experiments, Figure 4.15 represents the number of errors as a function of the training data set size.




(a) **K2** (b) **MWST**

Figure 4.15: Number of incorrect arcs in function of the number of training data.

Once more we can see that the difference between the score are very small; the number of training point that should be used to avoid any fear of error is about 1300, over this number all the algorithms give a correct answer.

On one hand, we can notice a 'step' with the **K2** algorithm, indeed between 200 and 900 training data the number of error with resulting network is constant: 2. On the other hand, with the **MWST** algorithm we notice that the number of errors is less constant with less than 1000 values but if we train the network with more than 1300 values, the resulting network is exactly the one we are looking for. The mean of error is

higher in the **K2** algorithm(2.3 errors with this algorithm and 2 for the **MWST**) but the standard deviation is smaller for the **K2** , results are less (2.7 for the **MWST** and 2.4 for the **K2**).

From these experiments we can conclude some points:

- The **K2** algorithm strongly depends on the order; if the node are ordered correctly the results are pretty good,but otherwise the results are awful.
- When the correct structure is a tree, **MWST** becomes more interesting, since it does not require any knowledge of node ordering and gives almost the same results as the **K2** algorithm even faster.
- The scoring function used does not seem to affect the resulting structure a lot.

# Chapter 5

# Experiment on Real Data

## 5.1 Description

We are now working with the *real data* (i.e. data collected from **GRiST** ) we assume the structure to be a tree and to order the node, we will use the order of the questions of Figure 2.3.

The dataset consists of 308 anonymised answers of **GRiST** in an Excel sheet. We are focused on the suicide sub-domain so we will use only answers associated with Figure 2.3 and *General information.* As mentioned earlier, some special answers are encountered: 'don't know' and the absence of answer; these are represented respectively by 99 and $\#NUL!$.

We performed some preprocessing on this data in order to be able to use the toolbox.

- First of all, we will add 2 to all the data values so the answers will be now scaled from 1 to 11 for questions 3 to 10 and from 2 to 3 for questions 1 and 2.
- We note that questions number 1 and 2 are in *reverse* scale: indeed in the questionnaire 0 (so 2 in our new data sheet) represented 'YES' and 2 meant 'NO'. We swaped these answers in order to be coherent with the rest of the questions: low value means unlikely and high value means probably.
- Finally, the toolbox treat $\#NUL!$ answer as $Nan$ (e.g. Not a number) and all the algorithm do not work in the presence of NaNs. In order to deal with this issue, we considered the absence of an answer as an absence of knowledge. Thus such values were treated as 'don't know' and we regrouped these values in one category represented by number 1. We discussed before of the thin line between 'NO' and 'don't know' that's why we have chosen to assign a small value to that answer.

Another important thing to notice is that for a network with 10 nodes, the result of the previous Chapter show that to retrieve its structure requires much more than 308 data points if all nodes are binary. Nevertheless, here, nodes which will be associated

to each question could have between 3 answers for question 1 and 2, and 12 answers for the others.

From the results in the previous Chapter, we have seen that for sparse networks (i.e. with few links) the differences among the scores are very small even when they exist. That is why we will use only Bayes' score for the **K2** algorithm and mutual information when we used the **MWST** algorithm. We will use the order of the questions, that means that some links should be present, indeed question 4 should only be answered if there was an answer to question 3; so there must be a link from node 3 to node 4. For the same reason there must be a link from node 2 to 3 and from node 8 to 9.

Our first task will be to select among the *General settings* questions those which are influencing the risk of suicide. To do so we will use the **ARD** algorithm.

In all this chapter, when it is not mentioned, the seed used was 14.

## 5.2 Selection of Variable

The **A**utomatic **r**elevance **d**etermination (**ARD**) algorithm, introduced in [13], is an algorithm to test the dependencies among variables. This test is done by training a classification model in a Bayesian framework.

The basic idea in **ARD** is to give each weight an hyperparameter:

$$p(w|\alpha) = \prod_i \mathcal{N}(w_i|0, \alpha_i^{-1}),$$

where $\alpha = \alpha_{i i=1}^{N}$ is a hyperparameter vector that controls how far away from zero each weight is allowed to go(it is the inverse of the variance). The hyperparameters $\alpha$ are trained from the data by maximizing the Bayesian 'evidence' $p(t|\alpha)$. The outcome of this optimization is that many elements of $\alpha$ go to infinity such that w would have only a few nonzero weights $w_j$ . This naturally prunes irrelevant features in the data.

In our case, we wanted to select variables among the *General settings* and for that we executed the **ARD** algorithm looked at the hyperparameters learned. The hyperparameters characterise the dependencies between the variable and the risk of suicide of the patient; the larger the hyperparameter *alpha*, is the less correlated the variables are. Here are the resulting hyperparameters found:

- For the question 'Sex', alpha = 7.24284,
- Question 'Marital status' alpha = 0.1703,
- Question 'Employment status' alpha = 0.5465,
- Question 'Accommodation' alpha = 0.9801,
- Question 'Household shared with' alpha = 0.0143.

So we can conclude from this experiment that the sex of the patient does not highly influence the risk of suicide but the other variable are really important. So we will now start the learning process with the question of Figure 2.3 plus four questions of the *General settings*:

1. 'Marital status',
2. 'Employment status',
3. 'Accommodation' and
4. 'Household shared with'.

## 5.3 Structure Learning and Prediction

The learning process was divided in 2 stages:

1. retrieve the **DAG** using the algorithms of Chapter 4,
2. once the **DAG** is found, we learn the parameters of the Bayesian network (i.e. the conditional probabilities).

The second step will not be detailed,since our forms is on the structure and the prediction. With the structure found we can try to predict the answer to the last question of **GRiST** (which was not shown in Figure 2.3): "*In your judgement, to what extent is the client at risk of suicide?*".

To do so we will perform some *inference* which means to estimate the probability of unknown ('missing' values) given some observation. We will observe the 9 first nodes, which means the 9 first answer plus the 4 answer of the general questions and try to predict the answer to the last question. A common algorithm to perform inference is the junction tree algorithm.

The junction tree algorithm is well-documented in [10]; the first idea in this algorithm is to work with potentials instead of probabilities. A potential is associated to a clique instead of a single node. A clique is a set of nodes that are all connected to each other. The original graph can be represented as a graph of cliques linked by their common nodes named the separator; such a graph is called a Cluster Tree. The 3 steps of the algorithm are as follows [1]:

1. Moralization:
   add links between parents of a shared child,
2. Triangulation:
   avoid loop of more than 3 nodes by adding a link between any two nodes in such a loop,
3. Construct the Cluster Tree as defined before with the corresponding potentials.

After that the potentials are updated so that each clique is coherent with every separator.

Nevertheless, as explained in [12] sometimes the junction tree algorithm is slow and may even not work. That is why we will use another engine existing in the **BNT** Toolbox: *likelihood_weighting_inf_engine* which is similar to the junction tree algorithm but gives better results.

The accuracy of the resulting network will be characterised by the number of correct predictions on a test dataset.

We started experimenting just with the data without forcing any links in the **DAG**; with these **DAG**s we will select the best seed to use among the 100 first seeds. To do so we performed the same experiment with all seeds: count how many prediction are correct. With the seeds found we will re-do the experiment to analyse the results. We will then perform the same process but with different **DAG**s: we forced some arcs to be present.

Then we reduced the number of possible answer to 4 categories (except for the last question) and performed the same experiments: firstly without forcing any links and then add some links.

We then reduced the number of possible answers for the risk of suicide to 4 categories as well.

Finally we tried removing the 4 questions of the *General settings* to see if the resulting structure and results change a lot.

### Experiments

The first experiment has been done with the whole dataset corresponding to Figure 2.3 plus the 4 other questions of *General settings.*

The resulting **DAG** are shown in Figure 5.1 for the **K2** algorithm and in Figure 5.2 for the **MWST** algorithm.

What appears on the first **DAG** is the absence of links between nodes associated to question 3 and question 4 of the suicide part. Otherwise we can notice that the other necessary links are found. The node associated to the risk of suicide is a leave in this **DAG**, which means that it depends on all nodes before it in the graph. Moreover, the nodes associated to the *general settings* are grouped together and only linked to the rest by one link.

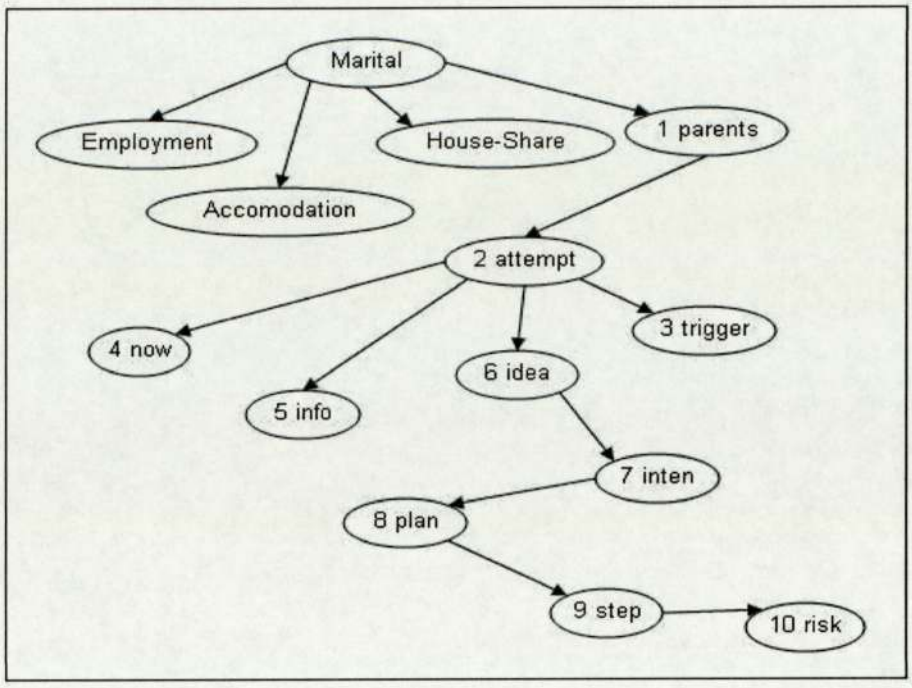



Figure 5.1: Resulting **DAG** for the complete dataset with the **K2** algorithm.

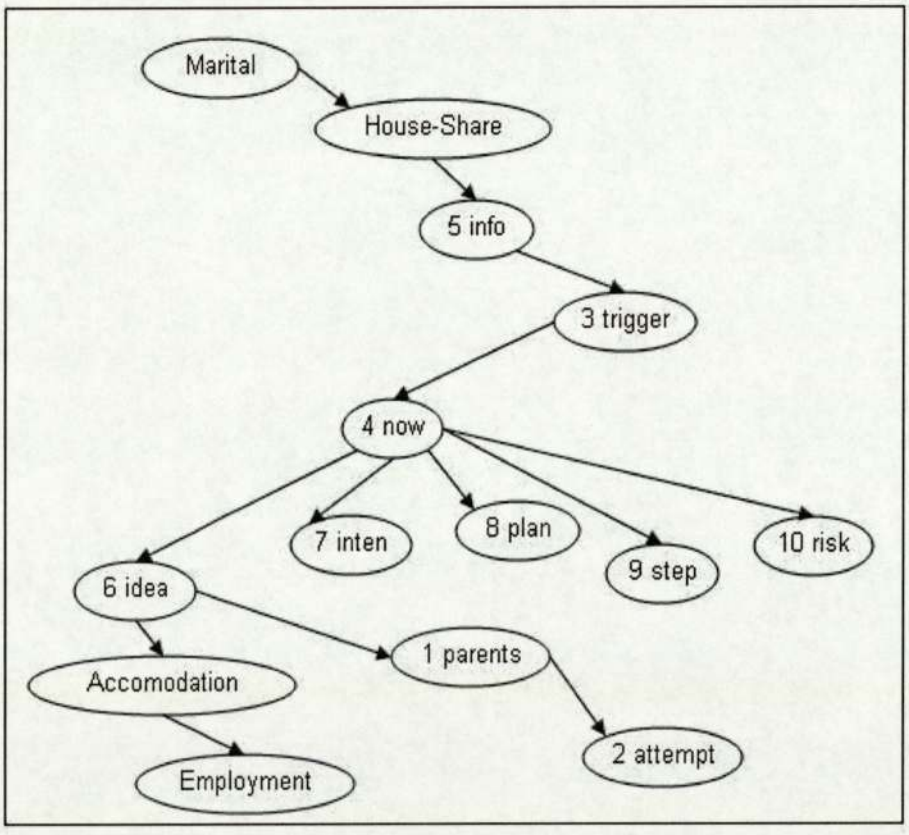



Figure 5.2: Resulting **DAG** for the complete dataset with the **MWST** algorithm.

On the second graph, we can highlight the lack of 2 necessary links ($2 \rightarrow 3$, $8 \rightarrow 9$). In this case the nodes associated to the *general settings* seems more linked to the other than in the previous graph.

With these two **DAG**s we have looked for the best seeds among seeds number 1 to seed number 100. To do so we used this algorithm:

```
for seed = 1..100
    for DAG=1..2
        learn the parameters with all the data
        for i = 1..308
            given the observation of the 13 first question of line i
            predict the risk of suicide
            if correct
                result(seed,j)+= 1
            endif
        endfor
    endfor
endfor
```

The choice of the seed will implies many change in the results. Basically, for a fixed seed the random generator is fixed too and will generate the same list of random number and in the previous chapter we highlighted the points where the algorithms use random numbers. As our dataset is not large, we perform the training and the test on the whole dataset.

Once this algorithm has run we just have to look at the maximum of $result(:, 1)$ to know which seed is the best for **DAG** associated to the **K2** and find the maximum of $result(:, 2)$ for the **MWST**.

For the first experiment, with the **K2** algorithm:

- the best result was 66 good predictions,
- the worst was 45 good prediction,
- the mean of correct answer is 50.5 with a standard deviation of 9.

For the **MWST** algorithm:

- the best result was 91 correct answers,
- the worst is 42 good prediction,
- the mean of correct answer is 74.84 with a standard deviation of 10.

The first thing to notice is that our prediction is really poor; if a network only predict 2 (i.e. no risk) it would have 160 good answer. Nevertheless, let us have a closer look to the result.

With seed number 66 with the **K2** algorithm we obtain 66 correct predictions; the confusion matrix is shown in Table B.1. Basically, the confusion matrix allows us to see which prediction was correct:

```
for each prediction p, associated to the real result r
    ConfusionMatrix(r,p)+=1
endfor
```

So $ConfusionMatrix(3,2)$ will represent the number of answers which are 3 that have been misspredicted to be 2. The diagonal of the confusion matrix, that show the correct predictions is: $(9, 25, 1, 1, 16, 3, 3, 1, 2, 4, 1, 0)$. That mean that only 25 2-answer, no risk at all, were correctly predicted. We also noticed that we try to predict "don't know" answer or the absence of answer so if we remove this case we had 52 correct predictions over 240 cases.

We also have performed the experiment with seed number 22 over the **DAG** found with the **MWST** algorithm; the confusion matrix is exposed in Table B.2 and if we remove the "don't know" answer the result is 67 correct predictions over 240 cases.

The second experiment has been done with the whole data corresponding to Figure 2.3 plus the 4 others question of *General settings*. In fact we just have forced the link that we had to, in order to be coherent with the questions(i.e. link $3 \rightarrow 4$ for the **K2** and links $2 \rightarrow 3$ and $8 \rightarrow 9$ for the **MWST**).

The resulting **DAG** are shown in Figure 5.3 for the **K2** algorithm and in Figure 5.4 for the **MWST** algorithm.

The only point to notice is that for the second graph, we had to remove one arc: the one between node 6 and node 1. Indeed if you look at the graph if you keep this arc we have a cycle :$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 6 \rightarrow 1$ so the graph is not a **DAG** anymore.

As in the previous experiment we have tested 100 seeds to find the best one; the results were:

- for the **K2** algorithm, 76 good predictions and
- for the **MWST** 77 correct predictions.

For the **K2** algorithm, the diagonal of the confusion matrix (see Table B.3) is: $(4, 32, 5, 1, 19, 4, 2, 2, 1, 3, 3, 0)$. The prediction is a little better than before, indeed without the "don't know" answer, we have 72 correct predictions over 240 cases.

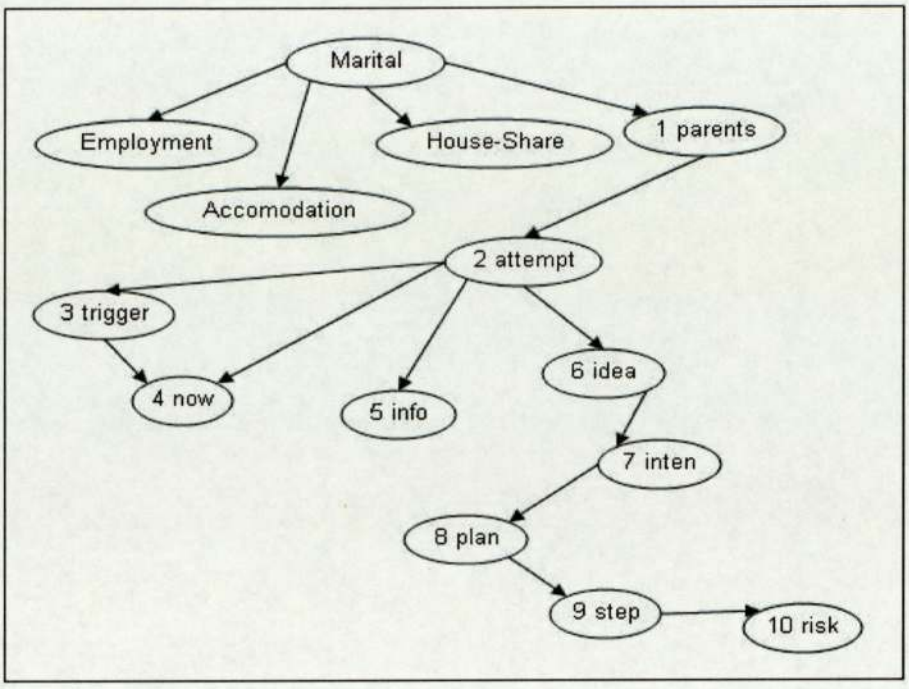



Figure 5.3: Resulting **DAG** for the complete data with the **K2** algorithm and some forced links.

With the **MWST** algorithm we finally obtain 78 accurate predictions, the diagonal of the confusion matrix (see Table B.4) is:
$(4, 36, 6, 2, 11, 7, 4, 2, 4, 2, 0, 0)$. 36 "no risk at all" were correctly predicted and we can see from the second column of the confusion matrix that many predictions are false because "no risk at all" is predicted very often.

In order to improve the results we tried to reduce the number of possible answer by grouping answers into 4 categories. Here is the list of the categories we have created for each question:

- Marital status:
  1. Single
  2. Married + Cohabiting
  3. Divorced + Widowed
  4. Other
- Employment status:
  1. Full Time
  2. Part Time
  3. Sickness benefit + Unemployed + Retired
  4. Other

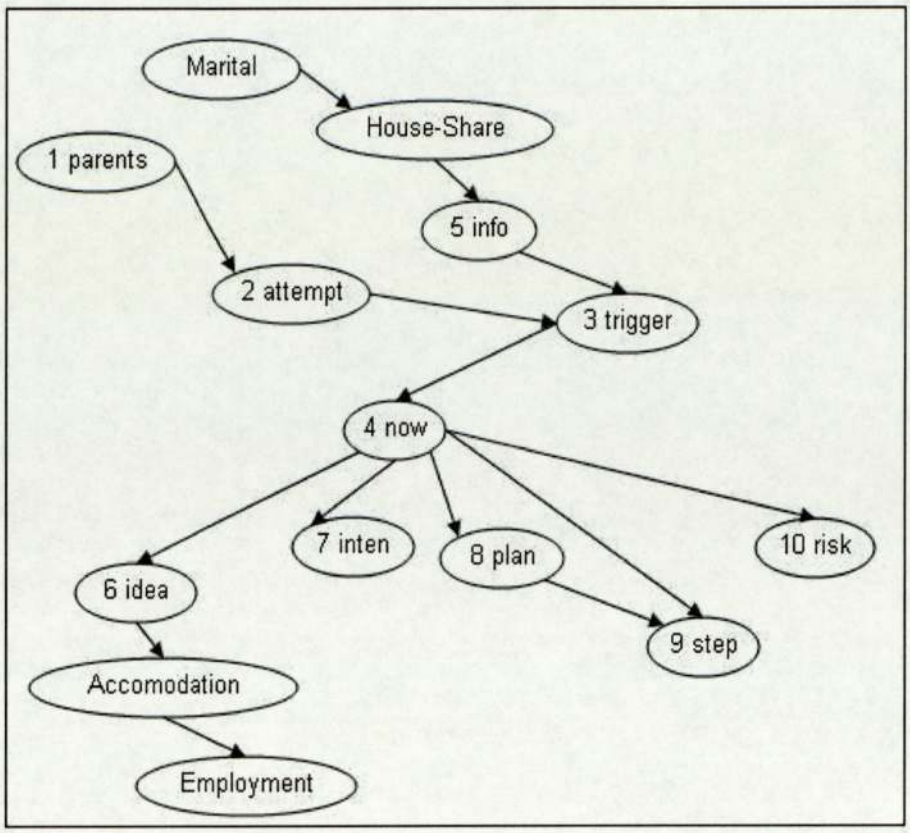



Figure 5.4: Resulting **DAG** for the complete data with the **MWST** algorithm and some forced links.

- Accommodation:
  1. Own home
  2. Rented home + hotel + Group home
  3. Homeless + Warden assisted
  4. Other
- Household shared with:
  1. Adult children + Spouse/Partner + Parents + Children + Other Relatives
  2. Friends
  3. Nobody
  4. Other
- Question 3 to 9 from the suicide part:
  1. Absence : 1
  2. very low : $2+3+4+5$
  3. average : $6+7+8$

4. high $9 + 10 + 11 + 12$

At first we kept the suicide risk on a 10 scaled answer and then, we also grouped this question. As before, we first learn the **DAG**s without forcing any links; the results can be found in Figure 5.5 for the **K2** algorithm and in Figure 5.6 for the **MWST** algorithm.

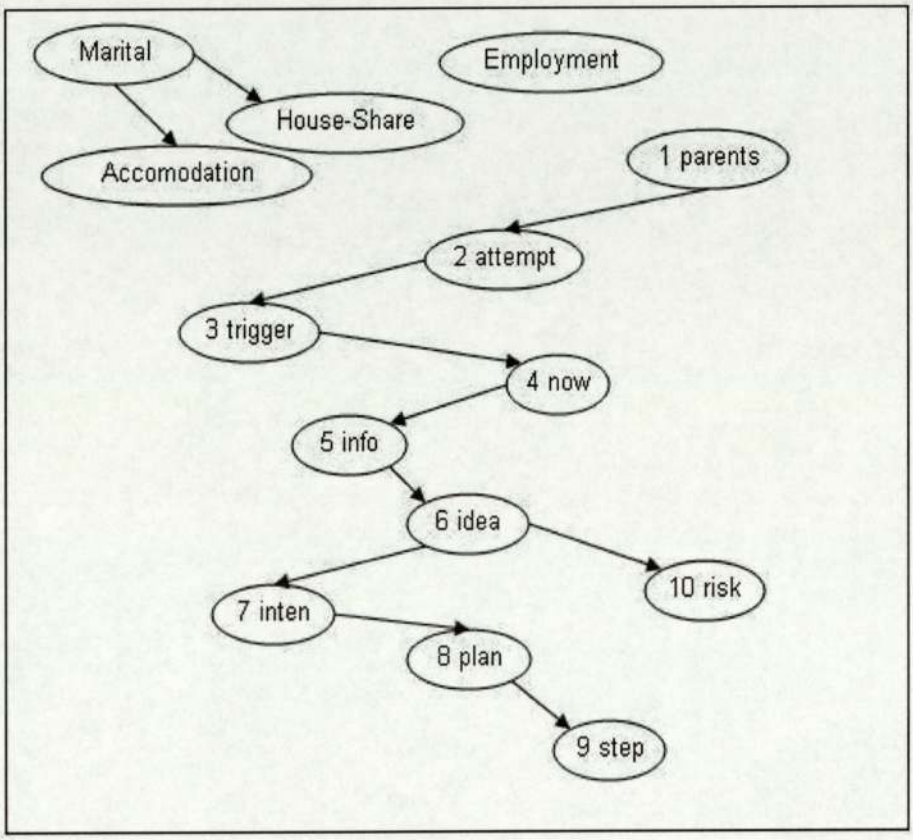



Figure 5.5: Resulting **DAG** for the reduced data with the **K2** algorithm.

The first thing to notice on the first **DAG** is the separation between the variables of *general setting* and the rest; the node employment is totally isolated. That means that the **K2** algorithm does not find any significant correlations between this variables and the rest. Another point to notice is that all the links that should be here are present, so this **DAG** is coherent with the order of the questions. On the other **DAG** , all the links are here as well but some of them are misdirected so we will have to modify them in the next step.

As usual we first find the best seed and then use that model to make predictions.

For the **K2** algorithm, the best result is 74 good prediction, the worst is 45; the mean of correct predictions is 58 with a standard deviation of 7.

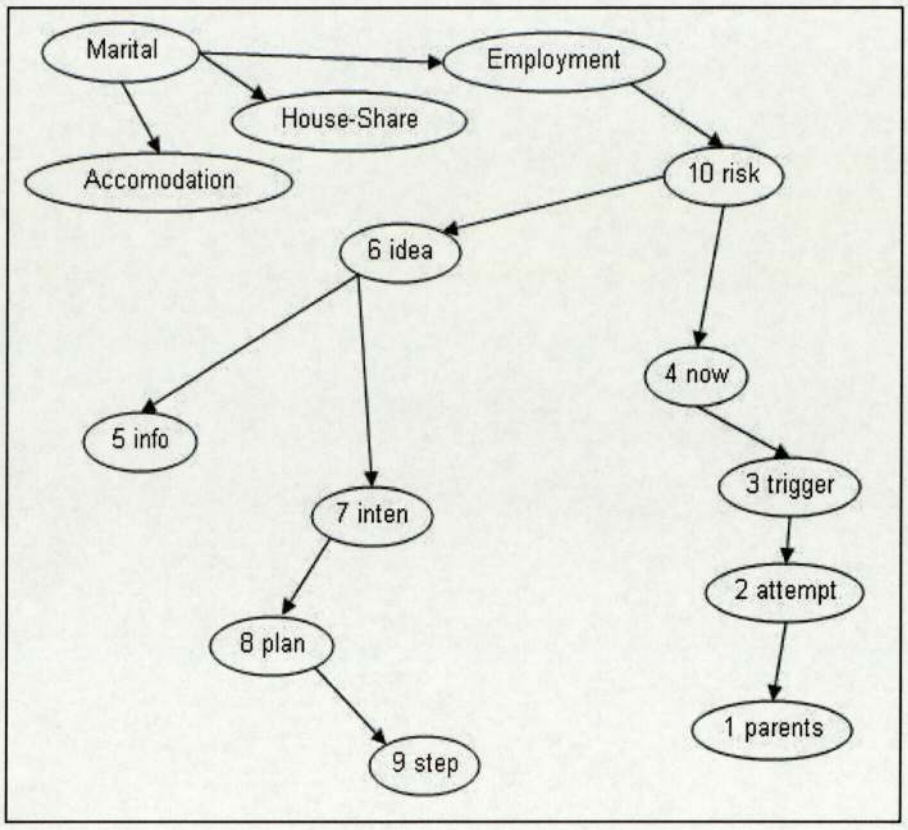



Figure 5.6: Resulting **DAG** for the reduced data with the **MWST** algorithm.

For the **MWST** algorithm,the best result is 75 accurate answer, the worst is 26; the mean of correct predictions is 50 with a standard deviation of 8.

For the experiment with the**K2** algorithm, the confusion matrix is represented in Table B.5; its diagonal is: $(23, 20, 2, 1, 10, 6, 5, 0, 4, 1, 1, 1)$. The final result are 51 good prediction for the **K2** algorithm. The results are so wrong that we will directly force some links for the **MWST**.

We did not need to force link for the first graph because we have seen that all links were present; nevertheless the second **DAG** is represented in Figure 5.7. The result was even worse than the previous experiment.

Indeed after having performed a search of the best seed for this **DAG** we launch the prediction with seed 70 which was the best but only 58 right prediction and if we remove the "don't know" answer the final results become 44 correct over 240 cases which is the worst result so far. The confusion matrix for this seed are represented in Table B.6. All these results are worse than before but it can be understood; indeed by grouping the possible answer we decrease the accuracy of the answer but we kept a 10 scale answer for the prediction so that's why the next experiment will group the last question into 4 categories as well.

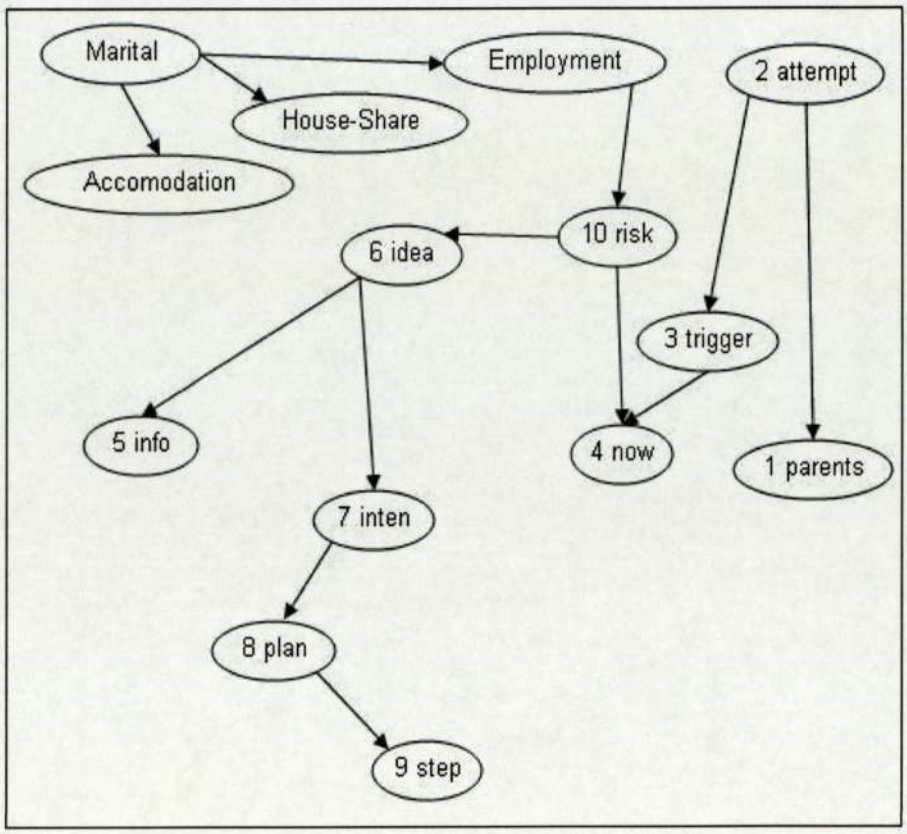



Figure 5.7: Resulting **DAG** for the reduced data with the **MWST** algorithm with some forced links.

The resulting **DAG** for the **K2** algorithm is the same as the one in Figure 5.5, for the **MWST** the result is in Figure 5.8.

As in the previous experiment some link are misdirected, for example there is a link between node 4 and node 3 which should be reversed. This orientation could be understood as an obligation of answer for node 3 where there is an answer to node 4 but we will force them in the other sense in the next experiment.

For the **K2** algorithm, the best result is **158** correct predictions, the worst is 115; the mean of correct predictions is 143 with a standard deviation of 10.

For the **MWST** algorithm, the best result is **158** correct predictions, the worst is 123; the average is 138 with a standard deviation of 7.

For the experiment with the **K2** algorithm, we found this confusion matrix:

$$\begin{pmatrix} 20 & 34 & 9 & 5 \\ 28 & 99 & 32 & 2 \\ 6 & 20 & 24 & 5 \\ 4 & 1 & 3 & 16 \end{pmatrix}.$$

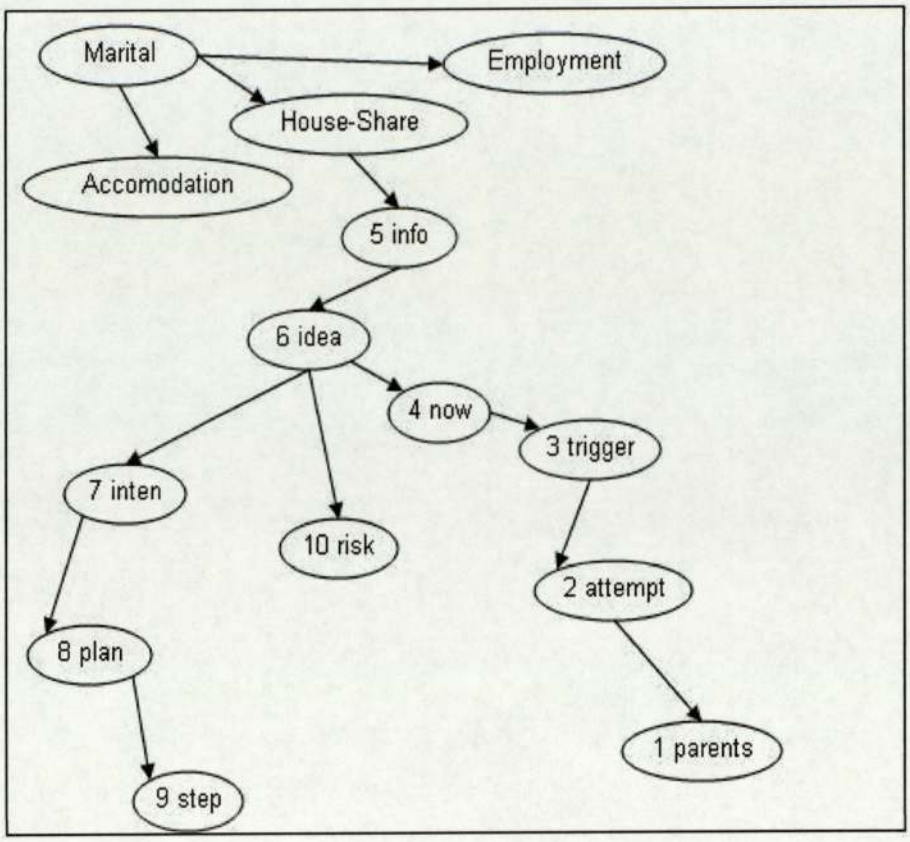



Figure 5.8: Resulting **DAG** for the totally reduced data with the **MWST** algorithm.

If we don't count the "don't know" answer we have 138 good prediction over 240 cases which is more than half good prediction. This result is still not higher than a network which only predicts 'no', which has 161 correct predictions; but 24 *average* (of the 55 average risk) and 16 of the 24 *very hight* risk have been correctly classified, 3 were classified as 'average' and only one is no risk.

With the same seed the **MWST** gives almost the same result; its confusion matrix is :

$$\begin{pmatrix} 14 & 40 & 12 & 2 \\ 34 & 92 & 34 & 1 \\ 4 & 24 & 22 & 5 \\ 2 & 4 & 5 & 13 \end{pmatrix}.$$

The final result for this **DAG** is 137 good predictions over 240 cases but we can notice that this time the number of 'very high risk' is less than before and 4 are ignored and classified as 'no risk'.

Final results with seed number 31 are almost the same (e.g. :135 for the **K2** and 140 for the **MWST**), so we will now perform another experiment where we have forced links in the correct sense in the **MWST** resulting **DAG**. The resulting graph is represented in Figure 5.9.

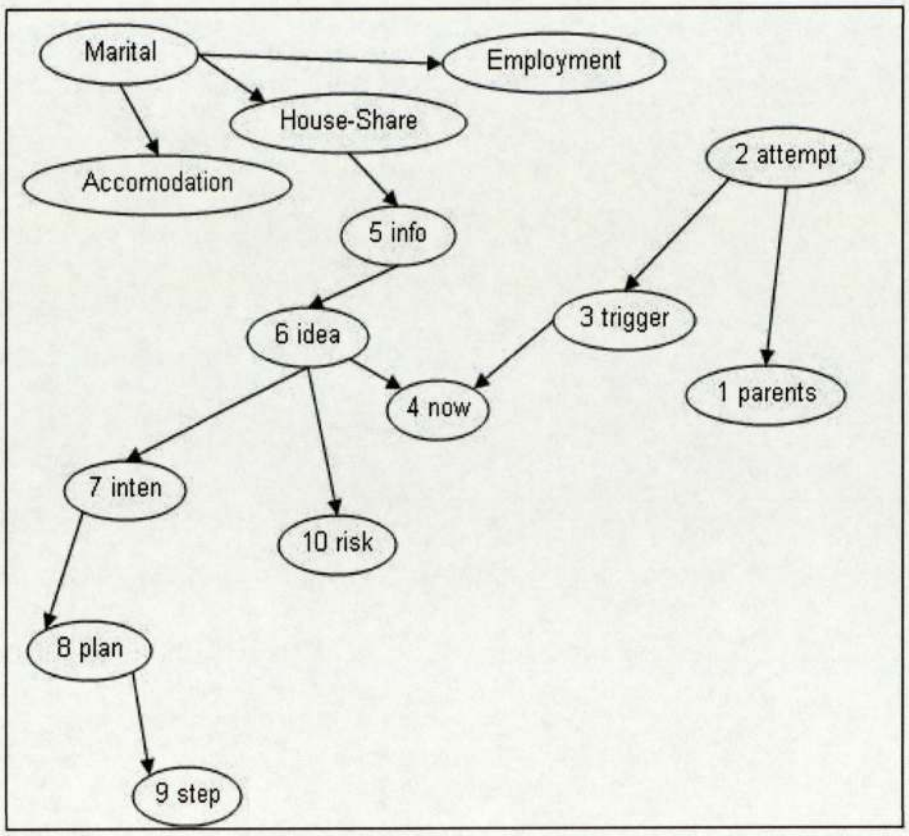



Figure 5.9: Resulting **DAG** for the totally reduced data with the **MWST** algorithm and forced links.

The best seed associated to this **DAG** is the 63 where the confusion matrix is :

$$\begin{pmatrix} 17 & 34 & 12 & 5 \\ 31 & 110 & 15 & 5 \\ 6 & 22 & 21 & 6 \\ 3 & 1 & 3 & 17 \end{pmatrix}.$$

One more time, if we remove the "don't know" answer from the prediction we have 148 good prediction over 240 cases and if we look closer to the confusion matrix we can see that the majority of the error are between very low risk and average risk. The high risk are well predicted only 7 are misclassified and only 1 is detected as 'no risk'.

Finally, in order to see the benefits of the insertion of the 4 variables of the *general settings* we will perform a last experiment without these 4 nodes. We learned the **DAG**s with the two algorithms as usual and we directly forced some links. The resulting graphs are represented in Figure 5.10 for the **K2** algorithm and in Figure 5.11 for the **MWST**.

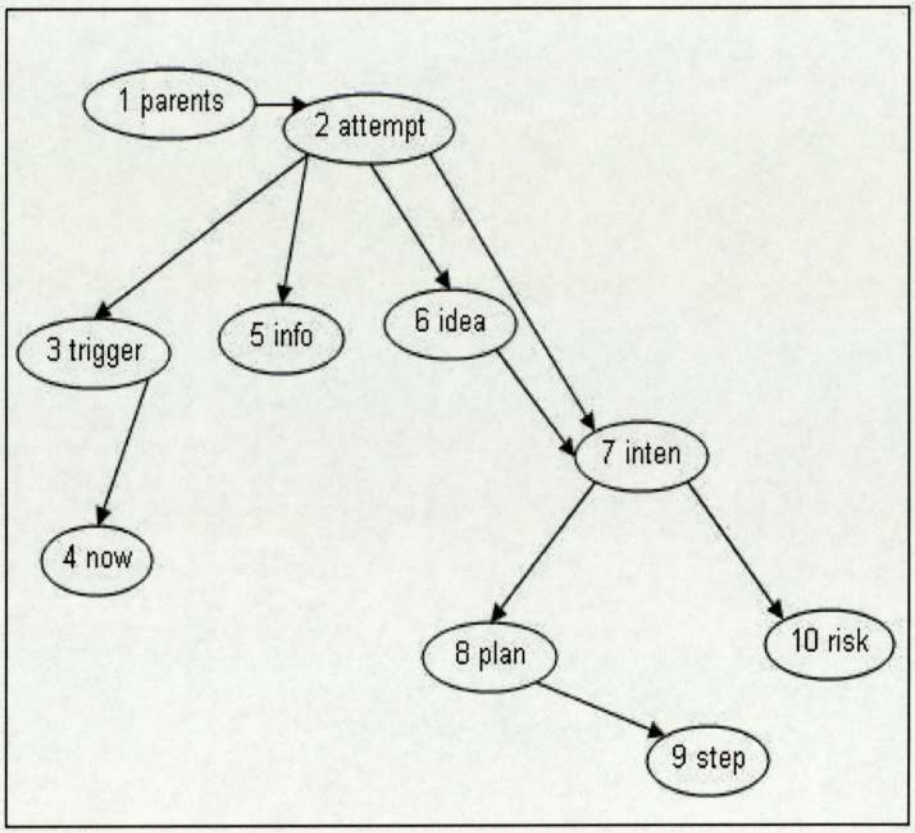



Figure 5.10: Resulting **DAG** for the reduced suicide data with the **K2** algorithm and forced links.

We found the best seed as usual for the 2 **DAG**s and performed predictions with the corresponding networks. We used the best seed for the **MWST**: seed 51; the results were worse than before. The final results are only 100 correct predictions for the **K2** algorithm and with the **MWST** here is the confusion matrix:

$$\begin{pmatrix} 20 & 18 & 20 & 9 \\ 26 & 65 & 54 & 16 \\ 19 & 11 & 22 & 4 \\ 3 & 6 & 11 & 4 \end{pmatrix}.$$

We have only 91 correct predictions. With the best seed for the **DAG** associated to the **K2** algorithm the result are even worst for the **MWST** (e.g. final result: 56 correct predictions). The confusion matrix associated to the **K2** **DAG** is :

$$\begin{pmatrix} 18 & 35 & 14 & 0 \\ 29 & 111 & 16 & 5 \\ 9 & 25 & 17 & 5 \\ 3 & 2 & 4 & 13 \end{pmatrix}.$$

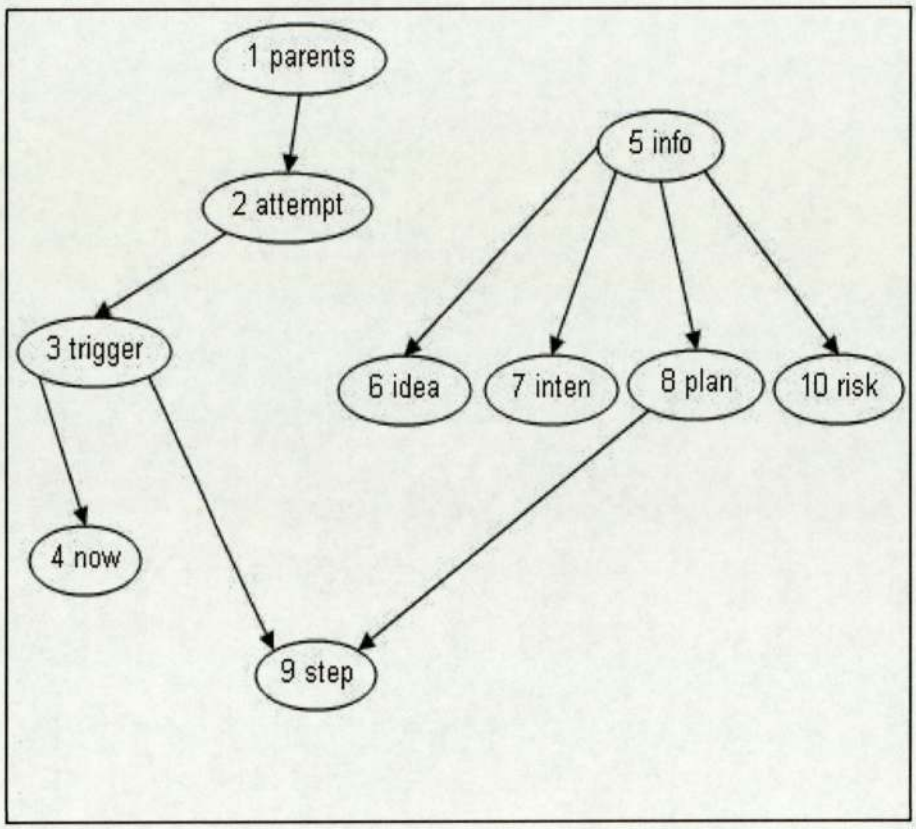



Figure 5.11: Resulting **DAG** for the reduced suicide data with the **MWST** algorithm and forced links.

These are close to the results we had with the 4 additional nodes; indeed we obtain 141 good predictions over 240 cases.

What we can conclude from all these experiments is that to have added some extra nodes gives us better results for the predictions but even with a diminution of the number of possible answers, prediction is not perfect at all: a network which will always answer "very low risk" will obtain better results than ours. In term of classification accuracy, some of our networks were quite accurate on the high risk class, which is the most important. The high variabilities of the results with different seeds is strong evidence that the dataset is not large enough to learn Bayesian network structure and parameters reliably.

# Conclusion

In the past ten years medicine has modernized a lot and is still evolving. The mental health part of the medicine is an area where there is a lot of scope for technological assistance.

The idea to use mathematics and more specifically Bayesian network to help experts doing prediction has a growing success nowadays; indeed expert systems are used more and more often in our society.

Given the increasing number of suicides and mental health problems in modern society it was reasonable to think about an expert system to help predicting risk of suicide: that is one main target of **GRiST** and also of that project.

In this thesis, we have used different algorithms to retrieve an unknown structure from the data. Basically, we have tried the **K2** algorithm and the **M**aximum **W**eight **S**panning **T**ree algorithm. Once we had figured out the structure, the next step had been to predict the answer of the last question of **GRiST** (i.e. to know how likely the patient is to commit suicide).

In theory, when someone does not know the node order a good way to find it is to use the tree given by the **MWST** algorithm to order the nodes and then apply a morel general structure learning algorithm. Thanks to some experiments we have seen that this idea does not in practice every time.

We faced a lot of technical problems because we used algorithms already coded and not always well documented; understanding why sometimes it suddenly crashed was not easy.

We tried many different approaches for preprocessing of the data:

- we first started with the original data associated to the suicide part of the questions of Grist and we add 4 questions selected from the *General settings*,
- we forced some links to be coherent with the question and
- we regrouped answers into 4 categories.

Given the small amount of data, we were able to construct different structures and predict with a different accuracies (i.e. from 16.6% to 61.6%) the possibility of suicide of a patient given a piece of information. Nevertheless, this results should be taken

with caution; indeed to predict the risk we had to group the possible answers into three categories so we can predict only if the risk is:

- less than 33% of risk,
- between 33% and 66% and
- more than 66%.

The fourth category was used to regroup the special answer :"don't know" or the absence of answer; such answer could not be predicted reliably.

In other word the results are not wrong but suffer from a lack of precision. The final point to notice is that every error in this domain are important: if a potential suicide committer is not "detected" the tool is flawed but if a "safe" patient is suspected to be suicidal he will have to pass through long and difficult exercise which could led him to mental disorder.

For sure, a larger dataset will be welcome and it will even be better if all questions were answered and to remove all the missing data which were really significant on our little dataset.

# Appendix A

# Listing

Here is the code: to generate the sprinkler network of Figure 4.1.

```
% SPRINKLERINIT     Generates the sprinkler network


clear all; rand('state', 14); randn('state', 14);

% Number of nodes
N = 4;

% The different nodes
C = 1; S = 2; R = 3; W = 4;

% The graph
dag = zeros(N,N); dag(C,[S R]) = 1; dag(S,W) = 1; dag(R,W) = 1;

discreteNodes = 1:N;
nodeSizes = 2*ones(1,N); % binary nodes

% label
label = {'cloudy', 'sprinkler', 'rain', 'wetGrass'};

% The network
bnet = mk_bnet(dag, nodeSizes,'names', label);

% The prior
bnet.CPD{C} = tabular_CPD(bnet, C, [0.5, 0.5]); bnet.CPD{S} =
tabular_CPD(bnet, S, [0.5, 0.9 0.5 0.1]); bnet.CPD{R} =
tabular_CPD(bnet, R, [0.8, 0.2 0.2 0.8]); bnet.CPD{W} =
tabular_CPD(bnet, W, [1.0 0.1 0.1 0.01 0.0 0.9 0.9 0.99]);


% Print the network
```

```
% coordonate of the nodes
[x,y]= make_layout(dag); draw_graph(dag,label,zeros(1,N), x, y);
```

# Appendix B

# Confusion matrices

$$\begin{pmatrix} 9 & 29 & 2 & 3 & 15 & 2 & 3 & 1 & 0 & 1 & 3 & 0 \\ 0 & 25 & 2 & 3 & 18 & 6 & 5 & 1 & 1 & 1 & 2 & 0 \\ 1 & 7 & 1 & 4 & 5 & 1 & 0 & 2 & 2 & 0 & 1 & 0 \\ 1 & 5 & 1 & 1 & 4 & 2 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 7 & 4 & 6 & 16 & 1 & 4 & 2 & 0 & 3 & 0 & 0 \\ 1 & 7 & 0 & 2 & 2 & 3 & 2 & 4 & 0 & 1 & 2 & 0 \\ 0 & 4 & 2 & 0 & 2 & 1 & 3 & 1 & 0 & 2 & 0 & 0 \\ 2 & 0 & 0 & 1 & 5 & 3 & 2 & 1 & 0 & 0 & 1 & 1 \\ 2 & 0 & 1 & 1 & 1 & 2 & 0 & 3 & 2 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Table B.1: Confusion Matrix for the first experiment with the **K2** algorithm.

$$\begin{pmatrix} 0 & 30 & 6 & 6 & 9 & 2 & 2 & 5 & 3 & 4 & 0 & 1 \\ 2 & 37 & 5 & 1 & 12 & 4 & 1 & 3 & 1 & 0 & 1 & 0 \\ 1 & 14 & 0 & 2 & 5 & 0 & 0 & 3 & 0 & 1 & 0 & 0 \\ 2 & 9 & 1 & 0 & 2 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 5 & 14 & 2 & 3 & 9 & 2 & 1 & 5 & 5 & 3 & 3 & 0 \\ 2 & 2 & 3 & 0 & 0 & 6 & 3 & 1 & 2 & 3 & 1 & 1 \\ 1 & 1 & 2 & 1 & 0 & 2 & 4 & 2 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 5 & 1 & 2 & 3 & 2 & 2 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 5 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 1 & 1 & 1 & 2 & 0 & 0 \\ 1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Table B.2: Confusion Matrix for the first experiment with the **MWST** algorithm.

$$\begin{pmatrix}
4 & 24 & 7 & 3 & 18 & 7 & 3 & 0 & 1 & 0 & 1 & 0 \\
7 & 32 & 4 & 2 & 14 & 8 & 5 & 0 & 0 & 0 & 0 & 0 \\
1 & 6 & 5 & 0 & 6 & 1 & 5 & 1 & 0 & 0 & 1 & 0 \\
1 & 3 & 2 & 1 & 4 & 3 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 17 & 1 & 0 & 19 & 9 & 1 & 3 & 1 & 1 & 0 & 0 \\
1 & 6 & 1 & 1 & 7 & 4 & 0 & 2 & 1 & 0 & 1 & 0 \\
3 & 5 & 0 & 1 & 3 & 0 & 2 & 0 & 0 & 1 & 0 & 0 \\
4 & 2 & 1 & 1 & 2 & 3 & 0 & 2 & 0 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 3 & 2 & 1 & 0 & 1 & 2 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 3 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 3 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0
\end{pmatrix}.$$

Table B.3: Confusion Matrix for the second experiment with the **K2** algorithm.

$$\begin{pmatrix}
4 & 27 & 9 & 5 & 10 & 4 & 4 & 4 & 1 & 1 & 0 & 0 \\
1 & 36 & 7 & 6 & 4 & 3 & 1 & 5 & 3 & 1 & 0 & 0 \\
0 & 11 & 6 & 1 & 4 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\
2 & 9 & 2 & 2 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
7 & 16 & 3 & 0 & 11 & 3 & 3 & 3 & 2 & 4 & 1 & 0 \\
4 & 6 & 0 & 0 & 2 & 7 & 2 & 0 & 1 & 2 & 0 & 0 \\
1 & 0 & 1 & 0 & 3 & 0 & 4 & 2 & 2 & 1 & 1 & 0 \\
0 & 2 & 0 & 1 & 5 & 0 & 1 & 2 & 1 & 2 & 0 & 1 \\
0 & 1 & 0 & 1 & 3 & 3 & 0 & 0 & 4 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 2 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}.$$

Table B.4: Confusion Matrix for the second experiment with the **MWST** algorithm.

$$\begin{pmatrix}
23 & 15 & 3 & 6 & 6 & 4 & 5 & 5 & 0 & 1 & 0 & 0 \\
14 & 20 & 10 & 2 & 14 & 5 & 0 & 2 & 0 & 0 & 0 & 0 \\
4 & 9 & 2 & 2 & 7 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
3 & 4 & 3 & 1 & 1 & 2 & 0 & 1 & 1 & 0 & 0 & 0 \\
5 & 15 & 8 & 2 & 10 & 1 & 2 & 2 & 3 & 0 & 0 & 0 \\
4 & 1 & 2 & 0 & 7 & 6 & 0 & 1 & 1 & 1 & 1 & 0 \\
1 & 0 & 1 & 1 & 3 & 2 & 5 & 0 & 2 & 0 & 0 & 0 \\
4 & 2 & 0 & 0 & 3 & 4 & 3 & 0 & 0 & 0 & 0 & 0 \\
3 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 4 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 3 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}.$$

Table B.5: Confusion Matrix for the third experiment with the **K2** algorithm.

$$\begin{pmatrix}
14 & 15 & 8 & 5 & 12 & 6 & 3 & 3 & 1 & 1 & 0 & 0 \\
14 & 21 & 4 & 2 & 8 & 6 & 5 & 4 & 2 & 0 & 0 & 1 \\
11 & 4 & 3 & 0 & 3 & 1 & 0 & 3 & 1 & 0 & 0 & 0 \\
3 & 2 & 2 & 1 & 4 & 0 & 2 & 2 & 0 & 0 & 0 & 0 \\
12 & 7 & 2 & 2 & 10 & 3 & 2 & 9 & 3 & 2 & 0 & 0 \\
3 & 4 & 1 & 1 & 5 & 2 & 3 & 1 & 2 & 2 & 0 & 0 \\
2 & 4 & 0 & 1 & 2 & 2 & 3 & 1 & 0 & 0 & 0 & 0 \\
2 & 4 & 1 & 2 & 3 & 0 & 0 & 1 & 0 & 1 & 2 & 0 \\
3 & 2 & 1 & 1 & 2 & 1 & 0 & 0 & 2 & 0 & 0 & 0 \\
0 & 2 & 3 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}.$$

Table B.6: Confusion Matrix for the third experiment with the **MWST** algorithm.

# Bibliography


[1] D. Barber. Probabilistic models. `http://www.ncrg.aston.ac.uk/~barberd`, *MSc Lecture Notes.*

[2] C. D. Buckingham, A. Adams, T. Chan, A. Davis, H. Gage, C. Mace, I. T. Nabney, and R. Picking. A decision support system for mental-health risk screening and assessment. 2002.

[3] C. D. Buckingham and T. Chan. Analysis of initial data set gathered by the galatean risk screening tool, GRiST. July 2003.

[4] C. D. Buckingham, T. Chan, and S. Watson. Developing a mental-health risk-screening tool. 2002.

[5] S. Brockie A. E. Adams C. D. Buckingham, G. Kearns and I. T. Nabney. Developping a computer decision support system for mental health risk screening and assessment. *Current perspectives in healthcare computing*, pages 189–194, 2004.

[6] C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE transaction information theory*, IT-14(3):462–467, 1968.

[7] G. F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347, 1992.

[8] D. Heckerman. A tutorial on learning with Bayesian networks. *Report MSR-TR-95-06, Microsoft Research, Redmond, Washington, 1995. Revised June 96.*, 1995.

[9] D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. In *KDD Workshop*, pages 85–96, 1994.

[10] M. I. Jordan. The junction tree algorithm. *CS281A/Stat241A: Statistical Learning Theory*, 2004.

[11] P. Leray and O. Francois. BNT structure learning package : Documentation and experiments. 2004.

[12] K. Murphy. How to use the Bayes net toolbox. `http://www.cs.ubc.ca/~murphyk/Software/BNT/usage.html`, 2005.

[13] R. M. Neal. *Bayesian Learning for Neural Networks.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1996.

[14] J. Pearl. Fusion, propagation, and structuring in belief networks. *Artificial Intelligence*, 29(3):241–288, 1986.

[15] J. Pearl and T. S. Verma. A theory of inferred causation. In James F. Allen, Richard Fikes, and Erik Sandewall, editors, *KR'91: Principles of Knowledge Representation and Reasoning*, pages 441–452, San Mateo, California, 1991. Morgan Kaufmann.

[16] R. W. Robinson. Counting unlabeled acyclic digraphs. *Combinatorial Mathematics*, 622:28–43, 1977.

[17] Bayesia SA. Bayesia market simulator. `http://www.bayesia.com/GB/produits/~bms/MarketSimulator.php`, 2001-2005.

[18] P. Spirtes, C. Glymour, and R. Scheines. Causation, prediction and search. *Lecture Notes in Statistics*, 81, 1993.

[19] L. C. L. van der Gaag, S. Renooij, C. L. M. Witterman, B. M. P. Aleman, and B. G. Taal. Probabilities for a probabilistic network: a case study in oesophageal cancer. *Artificial intelligence in medecine*, 25(2):123–148, 2002.

# Index