Learning Bayesian Network Structure from Data

Cui Hao (China)

Department of Probability and Statistics, Institute of Mathematics

Eötvös Loránd University, Budapest, Hungary

A thesis submitted for the degree of

MSc in Mathematics

2018. May

Declaration

I herewith declare that I have produced this paper without the prohibited assistance of third parties and without making use of aids other than those specified; notions taken over directly or indirectly from other sources have been identified as such. This paper has not previously been presented in identical or similar form to any other German or foreign examination board.

The thesis work was conducted under the supervision of Prof. Pröhle Tamás at Eötvös Loránd University.

Budapest, May, 2018

Abstract

Bayesian networks (BNs) are graphical representations of the probabilistic relationships among a set of random variables. They are effective graphical models to handle randomness and uncertainty of the real world. Over the past decades, Bayesian networks have gained increasing interests and been widely applied in various areas.

The study developed in this thesis focuses on the score-based approach for identifying the Bayesian network structure from data. We start with a brief introduction of Bayesian network concepts and properties, then provide an overview of Bayesian network structure learning algorithms from data. We focus on the particular task of structure learning from fully observed data using a search-and-score paradigm. We discuss different scoring functions and their implications, and survey the literature on existing structure-learning algorithms. We analyze a real world dataset via the score-based algorithm in R environment with different scoring functions. We improve their prediction ability by using random restarts, specifying blacklist and whitelist and a tabu search approach. Lastly, we evaluate the sensitivity and specificity of the structures learned with different scoring functions. Finally, we give conclusions on Bayesian network structure-discovery task in the score-based algorithm domain and direct paths for future research.

Contents

1

Introduction

1.1 Motivation

In current society, many types of uncertainty and data types exist. Are real-life data related? e.g data of weather, data of ground wet, data of car accident, etc. Are there cause and effect relations? Real world data can have complex causal relationships and these relationships are often not perfectly observed due to the complexity of the environment. How to extract value from these data and build effective uncertainty models plays crucial role in predictions and decision-making. To understand the underlying network structure of complex relational data and the relations between different factors, efficient mathematical models should be established. Our focus is on learning the cause and effect relationships from observations of the environment, build causal models that can describe the situation, make predictions and decisions for the real world cases from the models.

Throughout this thesis, we will follow the notation of Koller and Friedman's Probabilistic Graphical Models [1] (abbreviated PGM from now on), a textbook containing a comprehensive overview of Bayesian networks. Bayesian network, originating from the work of Judea Pearl and his colleagues in the late 1980s, offers a framework for probabilistic reasoning. The Bayesian network is a powerful knowledge representation and reasoning tool under conditions of uncertainty [2]. Different from many other models that can cope with uncertainties between variables and contain graphs such as decision

trees, artificial neural networks and Markov networks, Bayesian networks exhibit the causal relationships. Bayesian networks examine the pattern of statistical dependencies in the dataset and attempt to conclude about the structural network of causal relationships that produced those dependencies. Bayesian networks are ideal models to represent and learn directed causal relationships. The benefits of the Bayesian network models include that:

(1) Bayesian networks have the ability to represent joint probability distributions, which can handle randomness and uncertainty of the real world through the established theory of probability [3]. Bayesian networks show dependencies between variables, so that they can predict certain variables via the BN structure and their joint probability distributions.
(2) Bayesian networks are effective graphical expressions for uncertain relationships of variables in the application domains. They display directional graphs, which can represent cause-effect relationships clearly and intuitively. Causal relationships are vital in data mining for the reasons that **(a)** they are beneficial for the understanding of expertise knowledge; **(b)** help to make accurate predictions.
(3) Bayesian networks are capable of representing many of the independencies in a domain through the structure, which is a directed acyclic graph.
(4) Bayesian networks have strong ability to cope with incomplete data.

Due to the above mentioned advantages, Bayesian networks have become effective tools for uncertainty analysis and thus widely applied in various fields. Bayesian Networks can be used for predictive modeling, pattern recognition, classification and regression. In the medical field, Bayesian networks have long been used for diagnosis, prognosis, and treatment selection. In artificial intelligence area, Bayesian networks have been used in natural spoken dialog systems, vision recognition, expert system. Recently, Bayesian networks have been used in data mining, search engine optimization, computational molecular biology, bioinformatics, and biological data integration [4].

1.2 Thesis Goals and Overview

In this thesis, we mainly explore the problem of learning Bayesian network structures from fully observed data, using a search-and-score paradigm. The goal of this thesis

is to recover the causal structure of Bayesian networks from a real-world dataset using score-based hill-climbing algorithm, demonstrate Bayesian network prediction abilities with different scoring functions as well as make several improvements. The remainder of this thesis document is structured as follows:

In **Chapter 2** we give an introduction of the concepts and basic properties of Bayesian network models.

In **Chapter 3** we present an overview of Bayesian network structure learning algorithms. Then we focus on the narrower task of learning the structure of a Bayesian network from fully observed data using a search-and-score approach, which turns the problem of finding the Bayesian network structure into an optimization problem.

In **Chapter 4** we illustrate the usage of statistical software R and relevant packages to learn Bayesian networks. We apply the structure learning algorithms in real-life dataset to find a structure that best represent the data by demonstrating the prediction ability. We discuss the advantages and disadvantages of hill-climbing learning methods and illustrate ways for improvement. We improve our results by random restarts, specifying whitelist and blacklist and using tabu search. Finally, we make a comparison between different Bayesian network classifiers that we learned and evaluate them using ROC and AUC analysis.

In **Chapter 5** we provide our conclusions and explore possible ideas for future research.

2

Bayesian Network Fundamentals

In this chapter, we first give some basic mathematical terminologies in probability and graph theory, mainly definitions and theorems. Then we introduce the fundamentals of Bayesian network and some important properties.

2.1 Preliminaries

Definition 2.1.1 (Conditional probability). Given two events A and B, from the σ-field of a probability space $(\Omega, \mathscr{F}, \mathscr{P})$, with $P(B) > 0$, the conditional probability of A given B is defined as the quotient of the probability of the joint of events A and B, and the probability of B:

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

Lemma 2.1.2 (Bayes' theorem). Given a probability space $(\Omega, \mathscr{F}, \mathscr{P})$ we get for all $A, B \in \mathscr{F}$, with $P(B) > 0$.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.2}$$

Definition 2.1.3 (Independence). Given a probability space $(\Omega, \mathscr{F}, \mathscr{P})$ and two events $A, B \in \mathscr{F}$, with $P(A), P(B) > 0$. A and B are independent (often written as $A \perp B$) if their joint probability equals the product of their probabilities: $P(A \cap B) = P(A)P(B)$. Or equivalently, $P(A \cap B) = P(A)P(B) \iff P(A) = \frac{P(A \cap B)}{P(B)} = P(A|B)$. And similarly,

$P(A \cap B) = P(A)P(B) \iff P(B) = P(B|A)$. Thus, the occurrence of B does not affect the probability of the occurrence of A, and vice versa.

Independence is a strong and useful property, however, in real life, we don't often encounter truly independent events. A more common property is conditional independence, where two events are independent given we have observed a third event.

Definition 2.1.4 (Conditional independence). Given a probability space $(\Omega, \mathscr{F}, \mathscr{P})$ and three events $A, B, C \in \mathscr{F}$, with $P(A), P(B), P(C) > 0$. A and B are independent given C if and only if $P(A \cap B|C) = P(A|C)P(B|C)$. Recall that from the definition of conditional probability (2.1), if A and B are conditionally independent given C, we obtain $P(A|B \cap C) = \frac{P(A \cap B \cap C)}{P(B \cap C)} = \frac{P(A \cap B|C)P(C)}{P(B \cap C)} = \frac{P(A|C)P(B|C)P(C)}{P(B|C)P(C)} = P(A|C)$. Thus we have

$$P(A|B \cap C) = P(A|C) \tag{2.3}$$

And similarly, $P(B|A \cap C) = P(B|C)$. We denote A and B independent given C as $A \perp B|C$.

Two events A and B are conditionally independent given a third event C precisely if the occurrence of A and the occurrence of B are independent events in their conditional probability distribution given C. In other words, A and B are conditionally independent given C if and only if given knowledge that C occurs, knowledge of whether A occurs provides no information on the likelihood of B occurring, and knowledge of whether B occurs provides no information on the likelihood of A occurring.

A set of rules governing statements of conditional independence can be derived from the basic definition. In the following lemma, the comma can be read as an "AND".

Lemma 2.1.5. Let A, B, C, D be random variables on the same probability space, then the following properties will hold:
Symmetry. $A \perp B \implies B \perp A$
Decomposition. $A \perp B, C \implies A \perp B$ and $A \perp C$.
Weak union. $A \perp B, C \implies A \perp B|C$ and $A \perp C|B$.
Contraction. $A \perp B|C$ and $A \perp C \implies A \perp B, C$.
Intersection. If the joint density of all variables with respect to the product measure is positive and continuous, we have $A \perp B|C, D$ and $A \perp D|C, B \implies A \perp B, D|C$.

Graph Terminology

This section states some basic concepts in graph theory which we will need to define the graphical part of Bayesian networks in the next chapter.

Definition 2.1.6 (Graph). A graph is an ordered pair $\mathscr{G} = (V, E)$ with V a finite set of vertices and E a set of edges, which are 2-element subsets of V.

Definition 2.1.7 (Directed graph). A directed graph $\mathscr{G}$ can be defined as an ordered pair that consists of a finite set V of nodes and an irreflexive adjacency relation E on V. The graph $\mathscr{G}$ is denoted as (V, E). For each $(x, y) \in E$ we say that there is an arc (directed edge) from node x to node y. In the graph, this is denoted by an arrow from x to y and x and y are called the start point and the end point of the arrow respectively. We also say that node x and node y are **adjacent** or x and y are **neighbors** of each other. x is also called a **parent** of y and y is called a **child** of x. By using the concepts of parent and child recursively, we can also define the concept of **ancestor** and **descendent**. We also call a node that does not have any parent a **root** node. By irreflexive adjacency relation we mean that for any $x \in V$, $(x, x) \notin E$, i.e., an arc cannot have a node as both its start point and end point [2].

If a graph contains only directed edges it is called a directed graph. In the same sense a graph with only undirected edges is an undirected graph. Intermediate variants (some edges directed, some edges undirected) are called partially directed. In this thesis, for graphical models we will focus on directed acyclic graphs.

Definition 2.1.8 (Directed acyclic graph). A directed acyclic graph (DAG) is a directed graph that contains no directed cycles.

2.2 Definition and Basic Properties

Bayesian networks are a class of probabilistic graphical models, which allow an intuitive representation of a set of random variables and their conditional independencies via a directed acyclic graph with a probability table for each node.

Definition 2.2.1 (Bayesian network). A Bayesian network $(\mathscr{G}, \mathscr{P})$ is a graphical representation of a probability distribution over a set of variables $U = \{X_1, X_2, ..., X_n\}$. It

consists of two parts:

(a) a BN structure $\mathscr{G}$, which is a directed acyclic graph (DAG) whose nodes are interpreted as random variables X_1, X_2,... X_n, while the edges between the nodes represent probabilistic dependencies among the corresponding random variables.

(b) a set of the *conditional probability distributions* ($CPDs$), one for each node/variable X_i, conditional on each value combination of the parents $P(X_i|Par_{\mathscr{G}}(X_i))$. The probability distribution of U is called the *global distribution* of the data.

The Bayesian network defines a joint distribution via the chain rule [1]

$$P(X_1, ...X_n) = \prod_i P(X_i|Par_{\mathscr{G}}(X_i)) \tag{2.4}$$

Lemma 2.2.2 Bayesian network is a legal distribution:

(a) $\mathscr{P} \geq 0$. $\mathscr{P}$ is a product of nonnegative CPDs.

(b) $\sum \mathscr{P} = 1$

A simple example of a Bayesian network in a discrete domain is shown in **Fig.** 2.1. It depicts a situation with five variables. Being at seaside causes people to go swimming, summer causes people to go swimming and eat ice cream, and going swimming makes people healthy. Let A denote "At Seaside", S denote "Summer", G denote "Go Swimming", E denote "Eat Ice Cream" and H denote "Healthy".

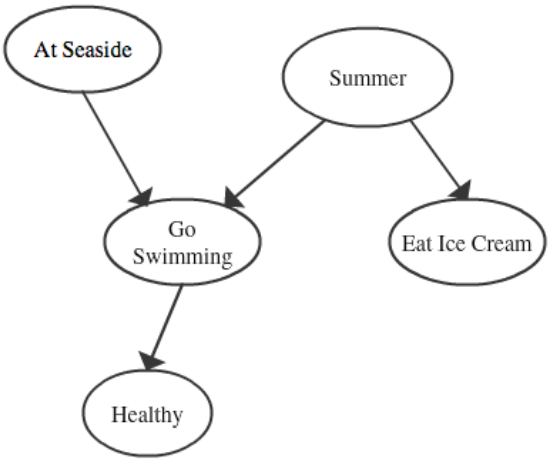

Figure 2.1: An example Bayesian network modelling summer activity. Chain rule for Bayesian networks.

The distribution of this Bayesian network can be defined as a product of factors using the Chain rule.

$$\begin{aligned}
\Sigma_{A,S,G,E,H} P(A,S,G,E,H) &= \Sigma_{A,S,G,E,H} P(A)P(S)P(G|A,S)P(E|S)P(H|G) \\
&= \Sigma_{A,S,G,E} P(A)P(S)P(G|A,S)P(E|S)\Sigma_H P(H|G) \\
&= \Sigma_{A,S,G,E} P(A)P(S)P(G|A,S)P(E|S) \\
&= \Sigma_{A,S,G} P(A)P(S)P(G|A,S)\Sigma_E P(E|S) \\
&= \Sigma_{A,S,} P(A)P(S)\Sigma_G P(G|A,S) \\
&= \Sigma_{A,S,} P(A)P(S) \\
&= 1
\end{aligned}$$

2.3 Flow of Probabilistic Influence

The direct edge from A to B means in general, influence can flow from A to B regardless of whether other variables are observed. In some cases, influence cannot flow between two variables.

Definition 2.3.1 ($V-$structure). A (part of) a Bayesian network graph structure where $A \rightarrow W \leftarrow B$ is called a $V-$structure.

Fig. 2.2 shows the situations when influence can flow between A and B and when influence cannot flow between A and B, a $V-$structure.

When can A influence B ?	
$A \rightarrow B$	Y
$A \leftarrow B$	Y
$A \rightarrow W \rightarrow B$	Y
$A \leftarrow W \leftarrow B$	Y
$A \leftarrow W \rightarrow B$	Y
$A \rightarrow W \leftarrow B$	N

Figure 2.2: Flow of probabilistic influence

Definition 2.3.2 (Active trail). Let $\mathscr{G}$ be a Bayesian network graph structure, C be a subset of the evidence nodes and $X_1 - X_2 ... - X_{n-1} - X_n$ be a trail between X_1 and X_n. An undirected path in $\mathscr{G}$ is called active trail for observed variables $C \subseteq \{X_1, ..., X_n\}$, if for every consecutive triple of variables X_{i-1}, X_i, X_{i+1} on the path
(a) $X_{i-1} \rightarrow X_i \rightarrow X_{i+1}$ and X_i is unobserved ($X_i \notin C$)
(b) $X_{i-1} \leftarrow X_i \leftarrow X_{i+1}$ and X_i is unobserved ($X_i \notin C$)
(c) $X_{i-1} \leftarrow X_i \rightarrow X_{i+1}$ and X_i is unobserved ($X_i \notin C$)
(d) $X_{i-1} \rightarrow X_i \leftarrow X_{i+1}$ and X_i or any of its descendants is observed
A trail $X_i - ... - X_n$ is active if it has no $V-$ structure $X_{i-1} \rightarrow X_i \leftarrow X_{i+1}$.

Fig. 2.3 shows the flow of probability influence given observed evidence between A and B.

When can A influence B given evidence about C ?		
	$W \notin C$	$W \in C$
$A \rightarrow B$	Y	Y
$A \leftarrow B$	Y	Y
$A \rightarrow W \rightarrow B$	Y	N
$A \leftarrow W \leftarrow B$	Y	N
$A \leftarrow W \rightarrow B$	Y	N
$A \rightarrow W \leftarrow B$	N (If W and all of its descendants are not in C)	Y (Either if W or one of its descendants is in C)

Figure 2.3: Flow of probabilistic influence given evidence

2.4 Independencies in Bayesian Networks

2.4.1 Conditional Independence

Definition 2.4.1.1 ($D-$seperation). Let X, Y, Z be three sets of nodes in $\mathscr{G}$. We say that X and Y are $d-separated$ in $\mathscr{G}$ given Z if there is no active trail in $\mathscr{G}$ between any node $x \in X$ and $y \in Y$ given Z. Notation: $d - sep_{\mathscr{G}}(X, Y|Z)$.

Use $I(\mathscr{G})$ denote the set of independencies that correspond to $d-$separation: $I(\mathscr{G}) = (X \perp Y|Z) : d - sep_{\mathscr{G}}(X, Y|Z)$. This set is also called the set of ***global Markov independencies***.

An example of $D-$separation is shown in **Fig.** 2.4. Consider the trail $A \rightarrow G \leftarrow S \rightarrow E$. If H is not observed, then $A \rightarrow G \leftarrow S$ is not active, thus the trail is not active. If H is observed, then the trail is active. If both H and S are observed, then the trail is not active, since S blocks the trail $G \leftarrow S \rightarrow E$.

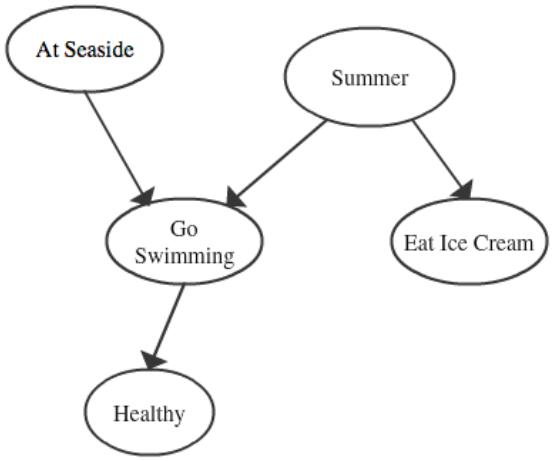

Figure 2.4: An example to illustrate D-separation

2.4.2 Independence and Factorization

Theorem 2.4.2.1 If $\mathscr{P}$ factorizes over $\mathscr{G}$, and $d - sep_{\mathscr{G}}(X, Y|Z)$ then $\mathscr{P}$ satisfies $(X \perp Y|Z)$.

Definition 2.4.2.2 (I-map). Let $\mathscr{G}$ be any graph associated with a set of independencies $I(\mathscr{G})$. Let $\mathscr{P}$ be a distribution over X. Let $I(\mathscr{P})$ be the set of independence assertions of the form $(X \perp Y|Z)$ that hold in $\mathscr{P}$. $\mathscr{G}$ is an ***I-map*** (independency map) for a set of independencies $I(\mathscr{P})$ if $I(\mathscr{G}) \subset I(\mathscr{P})$. Note that any independencies that $\mathscr{G}$ asserts must hold in $\mathscr{P}$ but $\mathscr{P}$ may have additional independencies that are not reflected in $\mathscr{G}$.

Theorem 2.4.2.3 If $\mathscr{P}$ factorizes over $\mathscr{G}$, then $\mathscr{G}$ is an *I-map* for $\mathscr{P}$.

Theorem 2.4.2.4 Let $\mathscr{G}$ be a Bayesian network structure over a set of random variables $\mathscr{X}$, and let $\mathscr{P}$ be a joint distribution over the same space. If $\mathscr{G}$ is an *I-map* for $\mathscr{P}$, then $\mathscr{P}$ factorizes over $\mathscr{G}$. Informally,
I-map $(X_i \perp NonDescendants(X_i)|Parents(X_i, \mathscr{G})) \Longrightarrow$ *Factorization* $P(X_1, ..., X_n) = \prod_{i=1}^{n} P(X_i|Parents(X_i, \mathscr{G}))$. This explains why BN factorize like this. An example can be given according to **Fig.** 2.4.

$$
\begin{aligned}
P(A, S, G, E, H) &= P(A)P(S)P(G|A, S)P(E|S)P(H|G) \\
P(A, E) &= \Sigma_{S,G,H} P(A)P(S)P(G|A, S)P(E|S)P(H|G) \\
&= \Sigma_S P(A)P(S)P(E|S)\Sigma_G P(G|A, S)\Sigma_H P(H|G) \\
&= P(A)(\Sigma_S P(S)P(E|S)) \\
&= P(A)P(E)
\end{aligned}
$$

2.4.3 Markov Blanket

Definition 2.4.3.1 (Markov blanket). Let V be a set of random variables, $\mathscr{P}$ be their joint probability distribution, and $X_i \in V$. Then a ***Markov blanket*** M_{X_i} of X_i is any set of variables such that X_i is conditionally independent of all the other variables given M_{X_i}. That is, $X_i \perp \{V - (M_{X_i} \cup X_i)\}|M_{X_i}$.

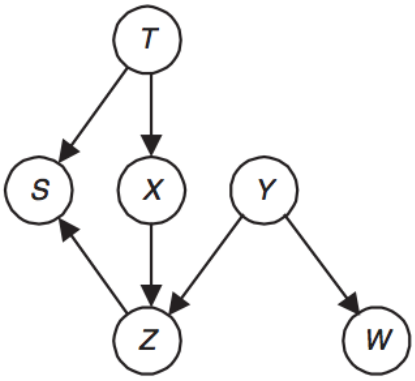

Figure 2.5: An example to illustrate Markov blanket

Definition 2.4.3.2 (Markov condition). Suppose we have a joint probability distribution $\mathscr{P}$ of the random variables in some set V and a DAG $\mathscr{G} = (V, E)$. We say that $(\mathscr{G}, \mathscr{P})$ satisfies the ***Markov condition*** if for each variable $X_i \in V$, X_i is conditionally

independent of the set of all its nondescendents given the set of all its parents. That is, $X_i \perp NonDescendants_{X_i} | Pa_{X_i}$.

Theorem 2.4.3.3. Suppose $(\mathscr{G}, \mathscr{P})$ satisfies the Markov condition. Then for each variable X_i, the set of all parents of X_i, children of X_i, and parents of children of X_i is a Markov blanket of X_i.

Suppose $(\mathscr{G}, \mathscr{P})$ satisfies the Markov condition where $\mathscr{G}$ is the DAG in **Fig.** 2.5. [5] Then due to **Theorem 2.4.3.3** T, Y, Z is a Markov blanket of X [5].

3

Bayesian Network Structure Learning

Learning, a term borrowed from expert systems and artificial intelligence theory, consists of learning graphical model structures and parameters. One important part of Bayesian networks learning is the structure learning (Korb and Nicholson, 2004; Koller and Friedman, 2009), finding the graph structure that encodes the conditional independencies present in the data and should ideally coincide with the minimal I-map of the global distribution [6]. The reason for structure learning is that for structure discovery, sometimes the domain expertise is not perfect when inferring network structure. Without knowing the structure and the probability table of a Bayesian network in advance, the goal of structure learning is to reconstruct the network based on the data available. Learning accurate structure is important since missing an arc lead to incorrect independencies while adding an arc lead to spurious dependencies and increase number of parameters. The structure learning of Bayesian network combines apriori knowledge and the sample to find the network structure which best fits the data. A fundamental assumption in structure learning is that the dataset contains independently and identically distributed (i.i.d.) [4] instances generated from an underlying distribution $\mathscr{P}$, which is induced by a Bayesian network $\mathscr{G}$.

In recent years, a number of computational methods have been developed for Bayesian network structure learning. Generally speaking, existing methodologies for constructing the Bayesian network structure from data can be categorized into three types: search

and scoring based algorithms (also known as Bayesian methods), constraint-based algorithms (also known as dependency analysis methods) and hybrid approaches; for an overview see Koller and Friedman (2009) and Scutari and Denis (2014) [7]. In the next section, we will introduce them in more details.

3.1 Score-based Algorithms

In this section, we will focus on one of the most popular methods of learning Bayesian network structures from fully observed data, the search-and-score paradigm. It is a heuristic optimization algorithm which ranks network structures with respect to a scoring metric, such as K2, BDe, AIC and BIC, to evaluate each candidate network structure, and try to search the optimal structure that fits best to the sample data. As the name suggests, a search-and-score method consists of two procedures, one is defining a scoring function that allows us to evaluate different networks. The second is a search strategy, an optimization procedure that allows us to identify the highest scoring structure over a typically large search space of possible network structures [4].

3.1.1 Scoring Functions

Search-and-score methods propose candidate structures, and evaluate them using a scoring function which measures how well that BN describes the data set $\mathscr{D}$. We want our scoring function to measure how well a structure fits the data. We also want it to penalize complex structures, because a simpler model makes inference more tractable. Scoring functions are commonly classified into two main categories: Bayesian scoring functions and information-theoretic scoring functions.

Bayesian scoring functions

The Bayesian score is one of several scoring functions satisfying the above mentioned criteria. Assuming a structure $\mathscr{G}$, its score is $Score(\mathscr{G}, \mathscr{D}) = Pr(\mathscr{G}|\mathscr{D})$, in other words, the posterior probability of $\mathscr{G}$ given the data set [3] [8]. Computation of the above can

be cast into a more convenient form by using Bayes' rule:

$$P(\mathscr{G}|\mathscr{D}) = \frac{P(\mathscr{D}|\mathscr{G})P(\mathscr{G})}{P(\mathscr{D})} \tag{3.1}$$

where the denominator is a normalizing factor that does not depend on the structure we are trying to evaluate. The goal of the score-based approach is to maximize the score that is assigned to each candidate BN. To maximize this we need only maximize the numerator, since the denominator does not depend on $\mathscr{G}$ [3]. The Bayesian score is then given by:

$$Score(\mathscr{G}, \mathscr{D}) = logP(\mathscr{D}|\mathscr{G}) + logP(\mathscr{G}) \tag{3.2}$$

Cooper and Herskovits [9] (1992) proposed K2 score, which is one of the earliest algorithms for learning Bayesian Network structure and it is a particular case of the *Bayesian Dirichlet* (BD) score [10]. K2 score requires a node ordering and maximum indegree as input. For each node, the algorithm greedily adds the parent node that gives the largest improvement in family score. When no additional parent increases the score, or when the maximum number of parents has been reached, the algorithm proceeds to select parents for the next node. Another score we are going to use is the *likelihood-equivalence Bayesian Dirichlet* (BDe) score which is derived by considering additional assumptions on likelihood equivalence and structure possibility [8], and its expression is identical to the BD expression.

Information-theoretic scoring functions

Information-theoretic metrics are based on compression. In this context, the score of a Bayesian network $\mathscr{G}$ is related to the compression that can be achieved over the data $\mathscr{D}$ with an optimal code induced by $\mathscr{G}$ [8]. The *minimum description length* (MDL) scoring function prefers simple Bayesian networks over complex ones, and it is rigorously defined as: $MDL(\mathscr{G}|\mathscr{D}) = LL(\mathscr{G}|\mathscr{D}) - 1log(N)|\mathscr{G}|$, where $|B|$ denotes the network complexity, N denotes the total number of instances in the data and LL denotes the *log-likehihood* score. A generalization of the MDL scoring function given by: $\phi(\mathscr{G}|\mathscr{D}) = LL(\mathscr{G}|\mathscr{D}) - f(N)|\mathscr{G}|$, where $f(N)$ is a non-negative penalization function. If $f(N) = 1$, we have the *Akaike Information Criterion* (AIC) scoring function, that is,

$AIC(\mathscr{G}|\mathscr{D}) = LL(\mathscr{G}|\mathscr{D}) - |\mathscr{G}|$. If $f(N) = 1log(N)$, we have the *Bayesian Information Criterion* (BIC) score based on Schwarz Information Criterion, which coincides with the MDL score. If $f(N) = 0$, we have the LL score [8].

In general, the scoring functions need to be decomposable over the network structure for efficiency purposes. The decomposability property allows for efficient learning algorithms based on local search methods. Moreover, when the learning algorithm searches in the space of equivalence classes of network structures, scoring functions must also be score equivalent [11], that is, equivalent networks must have the same score. In the application chapter of this thesis, we consider both Bayesian scores, such as K2, BDe and information-theoretic scores AIC and BIC/MDL.

3.1.2 Searching over Structures

Learning Bayesian networks is not straightforward. Cooper showed that the inference of a general Bayesian network is an NP-hard problem, and later, Dagum and Luby showed that even finding an approximate solution is NP-hard [8]. Apart from a scoring function, our next consideration in the score-and-search paradigm is how to find the high-scoring structure over the structure space. The standard methodology for addressing the problem of learning BNs is heuristic search, based on scoring metrics optimization, conducted over some search space. The search space includes the network structures and equivalence classes of network structures orderings over the network variables.

Major algorithms to search the space include greedy search algorithms, genetic algorithms and simulated annealing algorithms.

Greedy search algorithms

Greedy search algorithm such as hill-climbing (Chickering, Geiger, and Heckerman 1995), improves a network iteratively by performing local modifications: adding, removing, or reversing an edge. The search is started from either an empty, full, or possibly random network. Suppose we have some network structure, which is a DAG. We can define its neighborhood in DAG-space as all networks we can reach by applying an operator. The search operators include edge addition, edge deletion and edge reversal. We call it a legal modification when no cycles are produced during the process. In

each iteration, the algorithm computes the *delta score* (change in score) for each legal modification, and applies the one modification with the highest positive *delta* score [8]. The process stops when we reach the local maximum, where no change in the space of network gives rise to an improvement in the score. This learning approach is computationally efficient and, even though it does not guarantee an optimal result, many previous studies have shown that it obtains very good solutions. Hill climbing algorithms are particularly popular because of their good trade-off between computational demands and the quality of the models learned. The algorithm is shown as in **Table.** 3.1.

One pitfall of hill-climbing algorithm is that the learned structure might be local maxima rather than the global maxima. To alleviate this problem, hill climbing is often augmented with random restarts or tabu search (Bouckaert, 1995 [12]). By using random restarts, we take some number of random steps when get stuck at a local optimum, and then start climbing again. By using a Tabu lis, we keep a list of K steps most recently taken and do not revisit recently seen structures.

Table 3.1: An illustration of score-based algorithm

Greedy Hill Climbing Algorithm
1. Choose a network structure $\mathscr{G}$ over V, usually (but not necessarily) empty.
2. Compute the score of $\mathscr{G}$ Score($\mathscr{G}$), denoted it as $Score_{\mathscr{G}}$.
3. Set $Score_{max}$=$Score_{\mathscr{G}}$.
4. Repeat the following steps as long as maxscore increases:
(a) For arc addition, deletion or reversal not resulting in a cyclic network:
(1) compute the score of the modified network $\mathscr{G}^*$, $Score^*_{\mathscr{G}} = Score(\mathscr{G}^*)$.
(2) if $Score^*_{\mathscr{G}} > Score_{\mathscr{G}}$, set $\mathscr{G}^* = \mathscr{G}$ and $Score^*_{\mathscr{G}} = Score_{\mathscr{G}}$.
(b) update $Score_{max}$ with the new value of $Score_{\mathscr{G}}$.
5. Return $\mathscr{G}$.

Genetic algorithms

Genetic algorithms simulate natural evolution through the iterative selection of the "fittest" models and the hybridization of their characteristics (Larranaga et al., 1997). In this case the search space is explored through the crossover (which combines the structure of two networks) and mutation (which introduces random alterations) stochastic

operators [6].

Simulated annealing algorithms

Simulated annealing algorithms (Bouckaert, 1995 [12]) perform a stochastic local search by (always) accepting changes that increase the network score and, at the same time, allowing changes that decrease it (with a probability inversely proportional to the score decrease) [6].

3.2 Constraint-based Algorithms

Another way of learning BN structure is using constraints. Constraint-based algorithms are based on the seminal work of Pearl on causal graphical models and his *inductive causation* (IC) algorithm (Verma and Pearl 1991), which provides a framework for learning the DAG of a BN using conditional independence tests [6]. Tests in common use are the mutual information test (for discrete BNs) and the exact Student's t test for correlation (for GBNs). There are several assumptions for the conditional independence tests: Causal Sufficiency, Causal Markov, and Faithfulness (graphical separation and probabilistic independence imply each other) [3] [7]. An illustration of constraint-based algorithm is shown in **Table.** 3.2, S_{AB} represents d-separating set for nodes A and B.

Table 3.2: An illustration of constraint-based algorithm

Inductive Causation (IC) Algorithm
1. For each pair of variables $A, B \in V$ search for set $S_{AB} \subset V$ s.t. $A \perp B \mid S_{AB}$ and $A, B \notin S_{AB}$. Place an undirected arc between A and B if there is not such a set.
2. For each pair of non-adjacent variables A and B with a common neighbour C, if $C \notin S_{AB}$, set the direction of the arcs $A \rightarrow C$ and $C \leftarrow B$.
3. Set the direction of arcs which are still undirected by applying recursively the following two rules:
(a) if A is adjacent to B and there is a strictly directed path from A to B then set the direction of $A - B$ to $A \rightarrow B$.
(b) if A and B are not adjacent but $A \rightarrow C$ and $C - B$, then change the latter to $C \rightarrow B$.
4. Return the resulting (partially) DAG.

Constraint-based methods, such as GS, TPDA and sparse candidate, often start with a complete graph, and then carry out conditional independence (CI) tests to remove undesired edges as many as possible [13]. Constraint-based algorithms have two disadvantages in this category. One drawback is the exponential execution time in the number of variables of the domain. A more efficient algorithm is the PC algorithm whose efficiency comes from ordering the conditional independence tests from small to large. The algorithm is presented in detail in Spirtes et al. (1993) [3]. Another disadvantage is their poor robustness [3]. Small changes of the input can have large effects on the structure of BN and errors in the independence tests.

3.3 Hybrid Algorithms

Hybrid methods for Bayesian network structure learning combine the features of the constraint-based and score-based techniques. In the first step, independence tests are used to construct the skeleton of a Bayesian network to reduce the search space. In the second step, search-and-score techniques are applied to detect the directed acyclic graph that optimizes the scoring function of the initial Bayesian network [4]. Singh and Valtorta [14] integrated *conditional independence* (CI) tests to generate the ordering on the nodes and a Bayesian score to recover the network structures. Dash and Druzdzel [15] proposed to search the space of equivalent classes of essential graphs using a heuristic conventional constraint-based approaches and then to score with Bayesian metric. Acid and de Campos [16] developed a discrepancy-based scoring metric and a heuristic search strategy that emphasized the balance between model complexity and accuracy. Tsamardinos proposed a Max-Min Hill-Climbing (*MMHC*) algorithm, which combined methods in local learning and constraint-based to build an undirected network, and then perform a Bayesian-scoring greedy hill-climbing search to orient the edges [17]. Most recently, Hui Liu and Shuigeng Zhou developed a new hybrid method SAR (Separation And Reunion) [13], which decomposes the undirected independence graph for the full set of nodes in the separation phase, and learn small *DAGs* for the node set corresponding to each subgraph by applying a score-based method in the reunion phase.

4

Structure Learning Application

4.1 Introduction and Motivation

In recent years, learning Bayesian networks from data has become an increasingly active area of research. In general, they are built from a combination of data and expert opinions. Bayesian networks are quickly becoming the tool of many AI researchers for problems involving reasoning under uncertainty. They have been implemented in application areas such as medical diagnostics, prediction, anomaly detection, classification systems, software agents for personal assistants, decision automation (decision graphs), multisensor fusion, legal analysis of trials and many other tasks [18].

In this chapter, we will learn the structure of Bayesian networks from real world data. We will focus on a secondary school student alcohol consumption dataset. Our goal is to create a Bayesian network structure that describes the data well and achieve high prediction accuracy. Prediction is the process of calculating a probability distribution over one or more variables whose values we would like to know, given information (evidence) we have about some other variables. When a model is built from labeled training data, predicting outputs in the testing data is known as *supervised learning* [19].

In the following section, R environment and several useful packages are introduced. Bayesian networks are created and evaluated by using R packages. In this thesis, we use **bnlearn** package for the data from csv files and generate Bayesian networks which

represent dependencies and causal relationships between variables, using the score-based algorithms. The ROC analysis of our learned Bayesian network classifiers are implemented using **pROC** package.

4.2 R Environment and Relevant Packages

R is a free software environment which supports many statistical analysis methods for analysing and forecasting real life data, as well as graphical display. Our construction and analysis of Bayesian networks learned from the data will be implemented using R. In this section, we will introduce several useful packages that are used for Bayesian networks structure learning and evaluation.

bnlearn package

bnlearn [20] is an R package (R Development Core Team 2010) for learning the graphical structure of Bayesian networks, support for basic parametric and bootstrap inference, conditional probability queries and cross-validation (Koller and Friedman, 2009). **bnlearn** package focuses on the creation and manipulation of network structures, learning the structure of Bayesian networks, estimating its parameters and performing inferential procedures.

This package provides a free implementation of some structure learning algorithms along with the conditional independence tests and network scores, which are used to construct the Bayesian network. Both discrete and continuous data are supported [21]. Structure learning algorithms such as constraint-based, score-based (Hill-Climbing and Tabu Search) and hybrid algorithms are implemented. Available network scores in discrete case include the multinomial log-likelihood (Loglik) score, the Akaike Information Criterion score (AIC), the Bayesian Information Criterion score (BIC), which is equivalent to the Minimum Description Length (MDL), the logarithm of the Bayesian Dirichlet equivalent score (BDe), the logarithm of the K2 score (K2), and etc. Several functions for parameter estimation, parametric inference, bootstrap, cross-validation and stochastic simulation are also implemented [20]. The other suggested package, **Rgraphviz** (Gentry et al. 2010), can be installed from Bioconductor and is loaded along with **bnlearn**

if present. Advanced plotting capabilities are implemented on top of the **Rgraphviz** and **lattice** packages [6].

pROC package

The package **pROC** [22] provides tools for visualizing, smoothing and comparing *receiver operating characteristic* (ROC) curves and calculating the *area under the curve* (AUC).

4.3 Problem Statement

Alcohol influence our life badly in various aspects. For teenage students, alcohol can play a negative role in their study and social life, reducing their mental and physical abilities which can cause trouble for their future development. Students who consume large amount of alcohol tend to perform badly in academic studies, have poor decision-making skills, get involved with violence, take higher risk to have unprotected sex which can lead to sexually transmitted diseases and unwanted pregnancy [23]. Moreover, alcohol has many short-term and long-term health effects, drinking too much on a single occasion or over time can cause serious damage to our health. Teenage drinkers are more likely to face weight gain, disturbed sleep, headache, and are exposed to have other bad behaviours like taking illicit drugs. What's more, alcoholics are more likely to get injured or have accidents than non-drinkers, risking their lives at stake.

On the other hand, many factors can also contribute to the alcohol consumption of teenage students, such as students' family relationship, parents' education level and occupation, frequency of going out with friends etc. Adolescents living with both biological parents engage less frequently in heavy alcohol use than those living in any other arrangements. Living with a single mother is associated with less heavy drinking than living with a single father or with neither biological parent [24]. Teenage students' heath condition and academic performance is of unparalleled importance, since the younger generation is the hope and future of the development of the nation. It is the schools and families' obligation and responsibility to address significant concern on the secondary school students' growth.

Our aim is to learn the causal relations between alcohol consumption and other factors and build a classification system to predict those secondary school students who have the potential of consuming large amount of alcohol.

4.4 Data Preprocessing

In this section, we use a dataset describing secondary school students in Portuguese language course, found in the UC Irvine Machine Learning Repository. The dataset contains thirty-three discrete variables, stored as factors and integers with different levels in a a comma-separated csv file. This file can be loaded in R environment. Our dataset variables descriptions are shown in **Table.** 4.1.

To preprocess the data, we first create a new variable *Alcohol*, by combining the two attributes *Dalc* (daily alcohol consumption) and *Walc* (weekend alcohol consumption) using the equation $Alcohol = (5 \times Dalc + 2 \times Walc)/7$. The range of Alcohol is from 1 to 5. According to our calculations, 30.82% students consume alcohol greater than 2, 11.56% students consume alcohol greater than or equal to 3, thus 2 is a more suitable threshold, meaning those who consume weighted weekly alcohol greater than 2 are regarded as high alcohol consumers. Then we create a new variable *High* with categorical value *Yes* when *Alcohol* is greater than or equal to 2 which indicates that the student is a drinker, and categorical value *No* when *Alcohol* is less than 2 indicating not a drinker. Next, we transfer the age and the grades of the students to categorical values. The range of age is from 15 to 22. 27.89% students' age are greater than or equal to 18. We transform the data of age by whether their age reaches 18, with *Adult* indicating the age greater than or equal to 18 and *Teenager* indicating the age less than 18. Similarly, change the data of grades G1, G2 and G3 to categorical values *Low*, *Medium* and *Good* by their grades' distributions respectively.

4.5 Experimental Results

In this section, the R package **bnlearn** will be used to build the Bayesian network structure from our dataset. The package and its dependencies (the utils package, which is bundled with R) are available from CRAN. On constructing Bayesian networks from

Table 4.1: Dataset description

Attributes student-por.csv (Portuguese language course) datasets
1 school - student's school (binary: 'GP' - Gabriel Pereira or 'MS' - Mousinho da Silveira)
2 sex - student's sex (binary: 'F' - female or 'M' - male)
3 age - student's age (numeric: from 15 to 22)
4 address - student's home address type (binary: 'U' - urban or 'R' - rural)
5 famsize - family size (binary: 'LE3' - less or equal to 3 or 'GT3' - greater than 3)
6 Pstatus - parent's cohabitation status (binary: 'T' - living together or 'A' - apart)
7 Medu - mother's education (numeric: 0 - none, 1 - primary education (4th grade), 2 - 5th to 9th grade, 3 - secondary education or 4 - higher education)
8 Fedu - father's education(numeric: 0 - none, 1 - primary education (4th grade), 2 - 5th to 9th grade, 3 - secondary education or 4 - higher education)
9 Mjob - mother's job (nominal: 'teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at_home' or 'other')
10 Fjob - father's job (nominal: 'teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at_home' or 'other')
11 reason - reason to choose this school (nominal: close to 'home', school 'reputation', 'course' preference or 'other')
12 guardian - student's guardian (nominal: 'mother', 'father' or 'other')
13 traveltime - home to school travel time (numeric: 1 - <15 min., 2 - 15 to 30 min., 3 - 30 min. to 1 hour, or 4 - >1 hour)
14 studytime - weekly study time (numeric: 1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - >10 hours)
15 failures - number of past class failures (numeric: n if 1<=n<3, else 4)
16 schoolsup - extra educational support (binary: yes or no)
17 famsup - family educational support (binary: yes or no)
18 paid - extra paid classes within the course subject (Math or Portuguese) (binary: yes or no)
19 activities - extra-curricular activities (binary: yes or no)
20 nursery - attended nursery school (binary: yes or no)
21 higher - wants to take higher education (binary: yes or no)
22 internet - Internet access at home (binary: yes or no)
23 romantic - with a romantic relationship (binary: yes or no)
24 famrel - quality of family relationships (numeric: from 1 - very bad to 5 - excellent)
25 freetime - free time after school (numeric: from 1 - very low to 5 - very high)
26 goout - going out with friends (numeric: from 1 - very low to 5 - very high)
27 Dalc - workday alcohol consumption (numeric: from 1 - very low to 5 - very high)
28 Walc - weekend alcohol consumption (numeric: from 1 - very low to 5 - very high)
29 health - current health status (numeric: from 1 - very bad to 5 - very good)
30 absences - number of school absences (numeric: from 0 to 93)
These grades are related with the course subject:
31 G1 - first period grade (numeric: from 0 to 20)
32 G2 - second period grade (numeric: from 0 to 20)
33 G3 - final grade (numeric: from 0 to 20, output target)

data, we use nodes to represent attributes/variables and directed edges represent their causal relationships. Our expected learned structures are *DAGs*.

4.5.1 Blacklist and Whitelist

When applying Bayesian networks to real world problems, the directed edges learned by the algorithm may not correspond to the real life characteristics. For examples, in some of the learned structures, there are edges misdirecting to gender and age, meaning these features are the causes of gender and age, which is not true. In order to establish a Bayesian network structure which fulfills the causal relations in real life, we created a list of blacklist and whitelist based on the previously learned network structures and common knowledge (i.e. The grade of a student doesn't cause his/her age, or gender). Our constructed blacklist and whitelist are shown in **Table.** 4.2

Table 4.2: Blacklist and Whitelist

Blacklist	**Whitelist**
studytime → sex	guardian → High
High → sex	famrel → guardian
romantic → sex	age → higher
G1_Temp → sex	Medu → Mjob
romantic → age	Fedu → Fjob
G1_Temp → age	Fjob → famrel
G2_Temp → age	Mjob → famrel
higher → age	goout → absences
failures → age	age → guardian
	age → goout
	age → High
	sex → guardian
	sex → High
	goout → High
	Medu → famrel
	Fedu → famrel
	absences → failures
	High → G3_Temp
	High → failures
	High → absences

In **bnlearn** package, the learning algorithms support arc whitelisting and blacklisting.

Blacklisted arcs are never present in the graph. Arcs whitelisted in one direction (i.e. $A \rightarrow B$ is whitelisted but $B \rightarrow A$ is not) have the respective reverse arcs blacklisted, and are always present in the graph [20].

4.5.2 Learned Structures

The structures of the Bayesian network associated with this dataset are firstly learned with hill-climbing algorithm, implemented in the **hc** function, and stored in an object of class **bn**. We tried Bayesian scores BDe and K2 and information-theoretic scores AIC and BIC. In these cases, the error rates are relatively high, as shown in **Table.** 4.3 .

Table 4.3: Hill-Climbing algorithm prediction error

Scoring functions	Prediction error
K2 score	0.2824074
BDe score	0.2638889
AIC score	0.1990741
BIC score	0.2731481

We tried to improve our structure by using random restarts and tabu search. The tabu search algorithm is implemented in the **tabu** function, and also stored in an object of class **bn**. Hill-climbing and tabu algorithms result in a completely directed network. One way to visualize the network structure is to plot it either with the plot function or with **Rgraphviz** package which produces a better output for large complex graphs. Then we added blacklist and whitelist to our algorithm. Finally, we made a comparison of the network structures learned by hill-climbing and tabu algorithms with different scoring functions and with or without blacklist and whitelist. Some of our learned network structures are shown from **Fig.** 4.1 – **Fig.** 4.4.

4.5.3 ROC and AUC Analysis

The technique of *Receiver Operating Characteristic* (ROC), which is taken from signal detection theory, is a useful metric for visualizing, organizing and selecting classifiers [25] based on their performances. ROC graphs are commonly used in medical decision

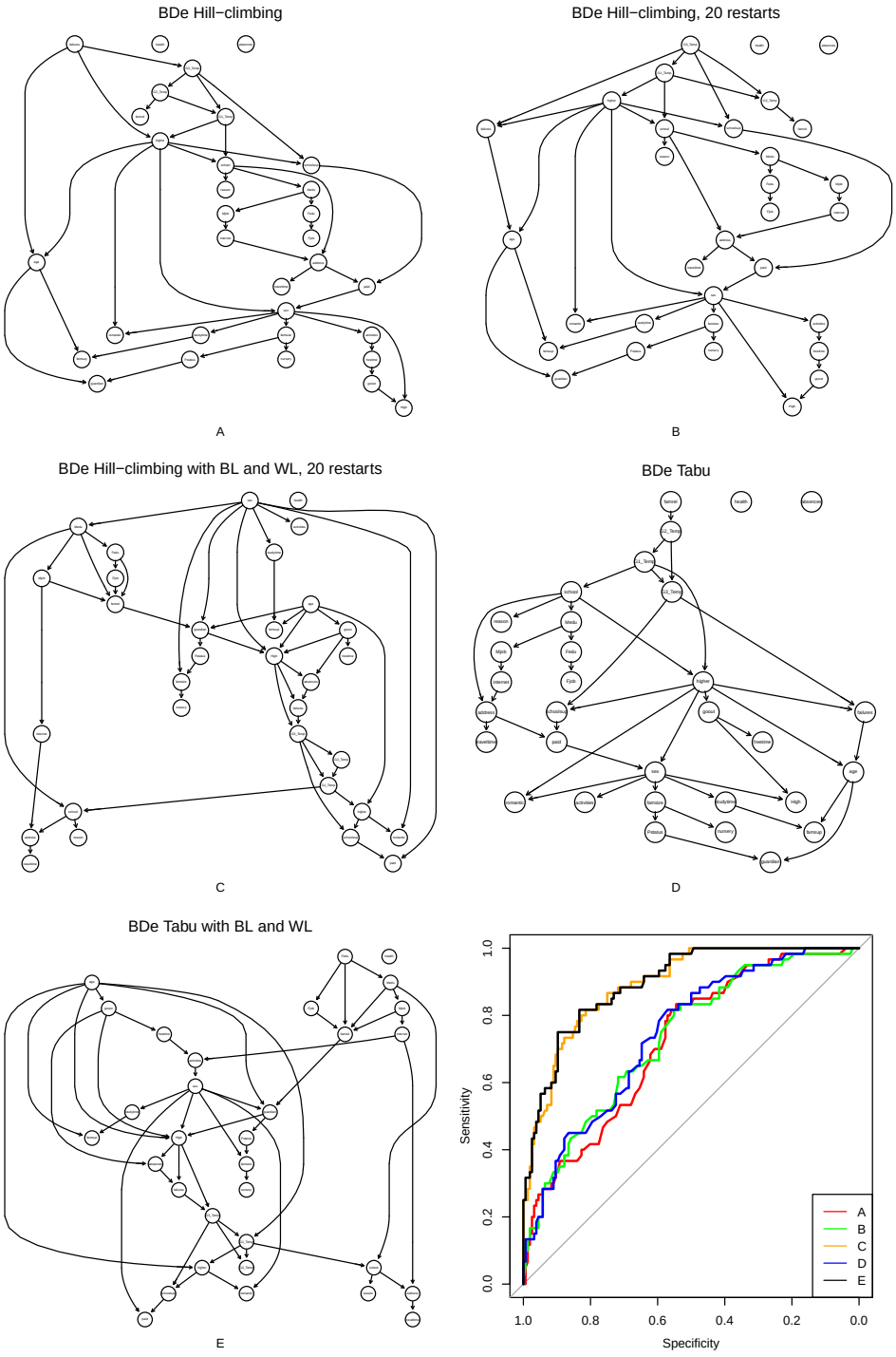

Figure 4.1: Bayesian Network built by BDe score

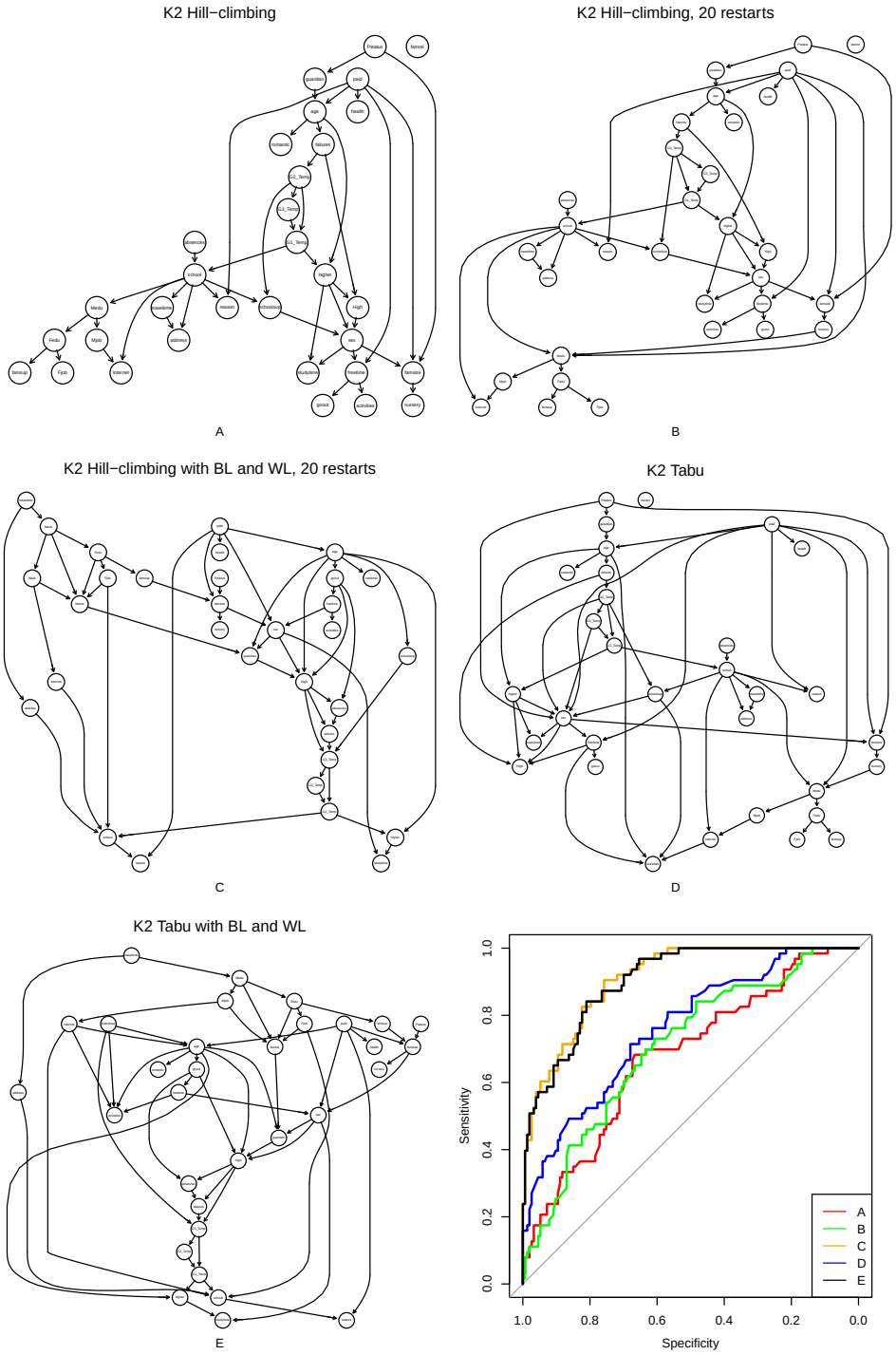

Figure 4.2: Bayesian Network built by K2 score

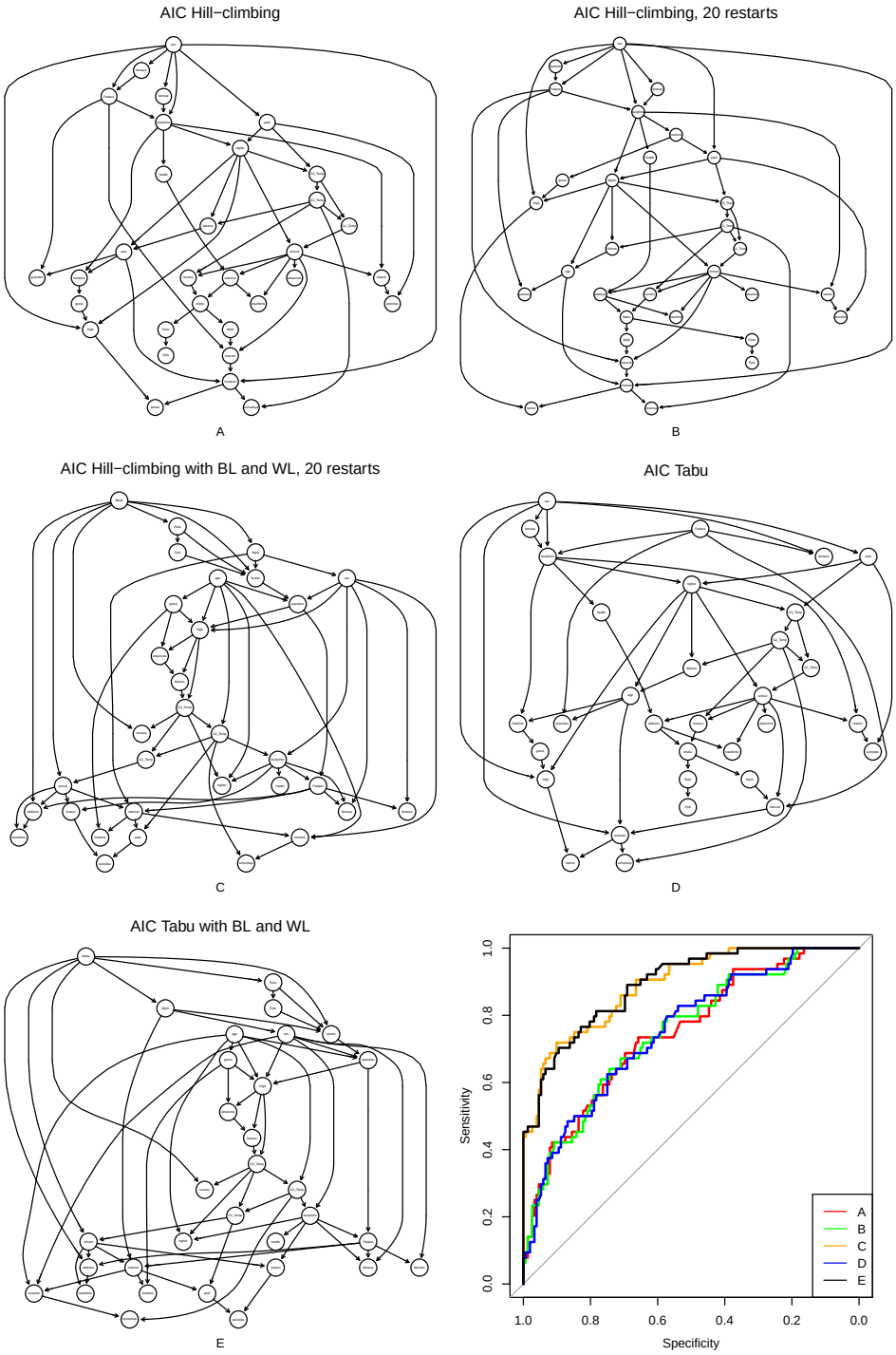

Figure 4.3: Bayesian Network built by AIC score

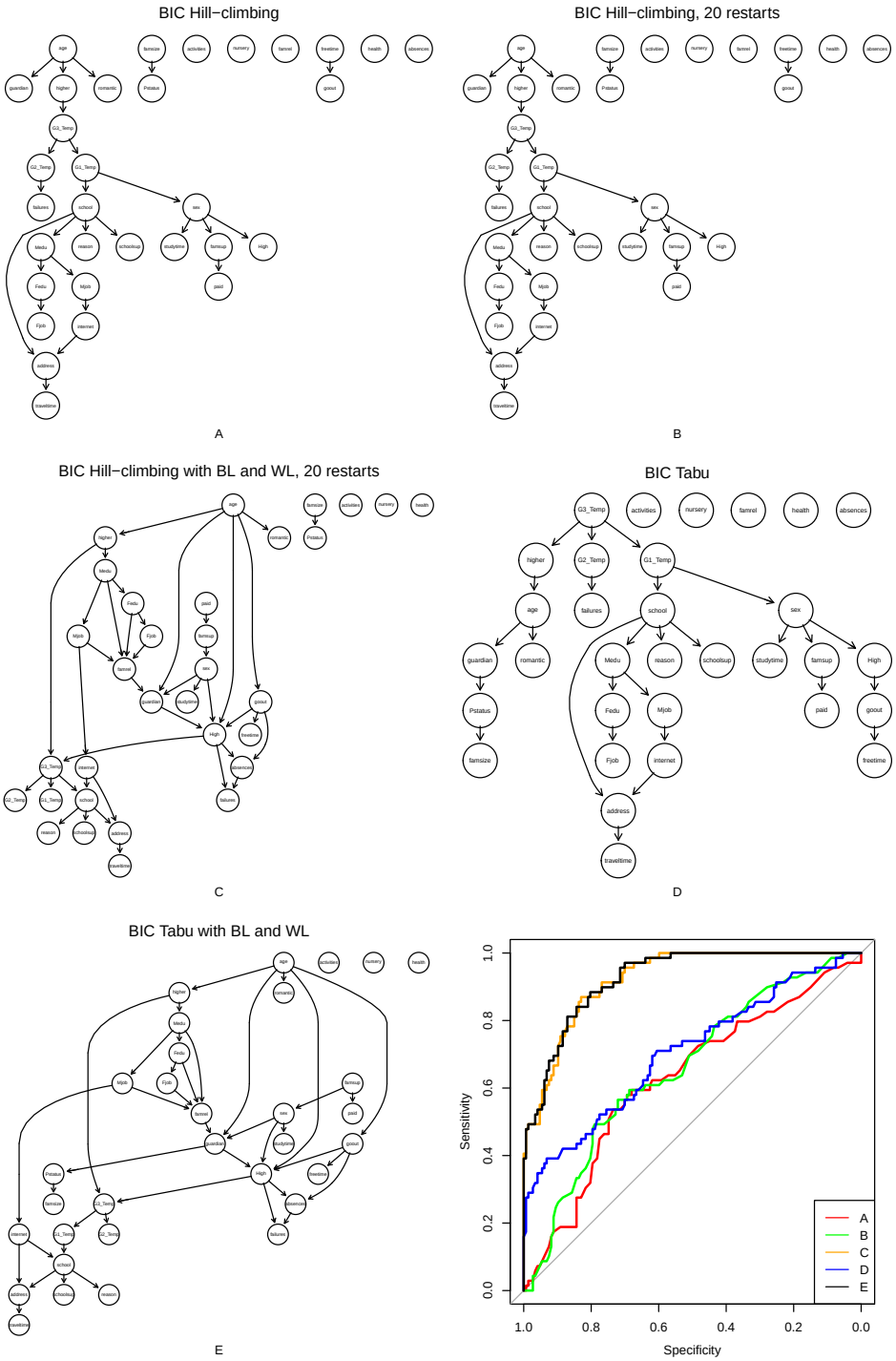

Figure 4.4: Bayesian Network built by BIC score

making, and in recent years have been used increasingly in the context of machine learning and data mining [25].

Table 4.4: Confusion matrix

Predicted condition	Condition positive	Condition negative
Positive	True positive (TP)	False positive (FP)
Negative	False negative (FN)	True negative (TN)

An ROC curve is a graphical approach for displaying the tradeoff between true positive rate (TPR) and false positive rate (FPR) of a classifier at various threshold settings of a diagnostic test. In an ROC curve, the true positive rate (TPR) is plotted along the y axis and the false positive rate (FPR) is shown on the x axis. The true-positive rate is also known as *sensitivity* and the false-positive rate can be calculated as (1-*specificity*). Mathematically, this can be expressed as:

$$TPR = sensitivity = \frac{numberofTPs}{numberofTPs + numberofFNs} \tag{4.1}$$

$$specificity = \frac{numberofTNs}{numberofTNs + numberofFPs} \tag{4.2}$$

$$FPR = \frac{numberofFPs}{numberofTNs + numberofFPs} = 1 - specificity \tag{4.3}$$

We can also say ROC curve demonstrates the tradeoff between sensitivity and specificity (any increase in sensitivity will be accompanied by a decrease in specificity). Each point along the curve corresponds to one of the models induced by the classifier [26].

Table 4.5: ROC curve interpretation

Value	Interpretation
TPR=0, FPR=0	Model predicts every instance to be a negative class
TPR=1, FPR=1	Model predicts every instance to be a positive class
TPR=1, FPR=0	The ideal model

There are several critical points along an ROC curve that have well-known interpretations, as are shown in **Table.** 4.5. A good classification model should be located as close as possible to the upper left corner of the ROC space, while a model that makes *random*

guesses (a record is classified as a positive class with a fixed probability p, irrespective of its attribute set) should reside along the 45-degree main diagonal, connecting the points (TPR = 0,FPR = 0) and (TPR = 1,FPR = 1) [26]. The closer the curve comes to the diagonal of the ROC space, the less accurate the test. The ROC curve for a random classifier always reside along the main diagonal since the TPR and FPR are identical.

A deterministic classifier produces a single point in ROC space, but a probabilistic classifier such as those considered in this work produces a curve. As stated by Provost and Fawcett [27], the benefit of ROC curves is that they illustrate the behavior of a classifier without regard to class distribution or error cost. ROC analysis is related in a direct and natural way to cost/benefit analysis of diagnostic decision making [28].

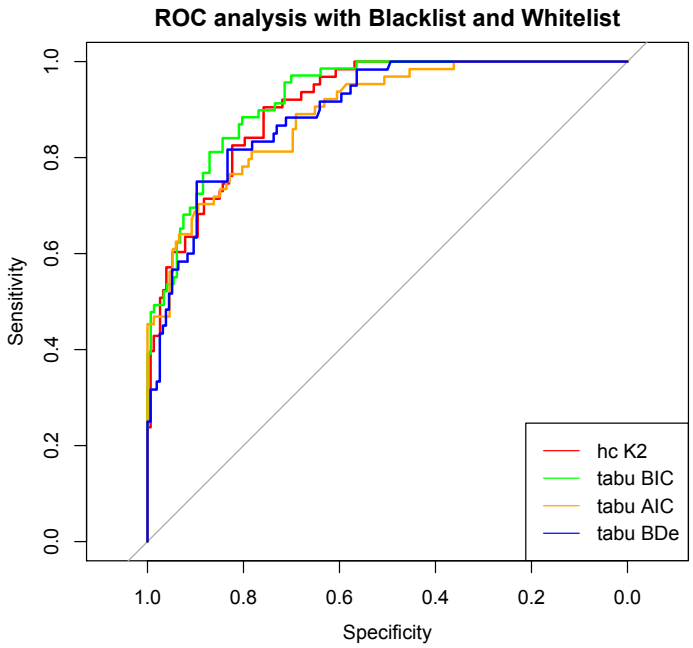

Figure 4.5: Sensitivity and specificity ROC curve analysis

The area under the ROC curve (AUC) provides another approach for evaluating which model is better on average. A perfect test has an area under the curve 1. If the model simply performs random guessing, then its area under the ROC curve would equal 0.5. A model that is strictly better than another would have a larger area under the ROC curve [26]. The area under the curve of different algorithms with different scores are

illustrated in **Table.** 4.6.

Table 4.6: Area Under the Curve (AUC) analysis

Algorithm score	K2	BDe	AIC	BIC
hc	0.6797	0.7200	0.7478	0.6272
hc, 20 restarts	0.7052	0.7324	0.7487	0.6582
hc, 20 restarts, BL and WL	**0.9101**	0.8927	0.8899	0.9232
Tabu	0.7606	0.7435	0.7506	0.7050
Tabu, BL and WL	0.9052	**0.8962**	**0.8917**	**0.9250**

4.6 Summary

Bayesian networks perform strong prediction ability and depict good directional graphical representations.

From the previous ROC graphs learned by Bayesian scores BDe, K2 and information-theoretic scores AIC and BIC respectively, we observe that the hill-climbing and tabu search algorithm with blacklist and whitelist have similar prediction abilities and perform dramatically better than those without blacklist and whitelist. For BDe score, K2 score and BIC score, the hill-climbing algorithm with random restarts and tabu search method (both without blacklist and whitelist) perform better than the classifiers learned by hill-climbing without random restarts, while for the AIC score, the difference is not obvious. In all cases, it improves the prediction ability better to create accurate blacklist and whitelist.

One disadvantage of Bayesian networks is that their learned directed edges can be wrong, thus lead to wrong causal relationships. Usually, expert opinions are important when building the network structure. Another disadvantage regarding heuristic search hill-climbing and tabu algorithms is that the running time will be very long when there are more variables and data.

5

Conclusions and Future Research

In recent years, Bayesian networks as graphical representations of the dependence relationships among a group of variables in an uncertainty domain, has seen growing applications in the real life problems. In this thesis, we focused on learning the structure of Bayesian networks from data, built Bayesian network classifiers and tried several approaches to improve their prediction abilities.

Firstly, we introduced the fundamental concepts in Bayesian networks, showed how probabilistic influence can flow on a network structure. Then, we presented three types of algorithms for Bayesian network structure learning, the score-based, constraint-based and hybrid algorithms. Next, we applied the score-based algorithm to a practical problem. We first built Bayesian network structures from the real world student alcohol consumption dataset using hill-climbing algorithm with four different scoring functions. Then we improved the prediction ability by specifying blacklist and whitelist, along with tabu search method. We evaluated different classifiers using ROC and AUC analysis and made a comparison between them.

5.1 Conclusions

From our previous work, we can derive several conclusions:

(1) Bayesian networks are efficient tools to depict causal relationships between variables via direct acyclic graphs.

(2) When learning the causal relations between different variables in the network and the prediction abilities of the classifiers, the setup of accurate blacklist and whitelist is essential.
(3) In general, tabu search algorithm performs better than greedy hill-climbing, since hill-climbing easily get stuck in local optima. Tabu search improves greedy hill-climbing by avoiding the learned structures in the tabu list created.
(4) In the application of this thesis, the best algorithm is tabu search method with blacklist and whitelist together with information theoretic scoring function BIC.

5.2 Future Works

During working on this thesis, we have arrived at useful conclusions and also discovered directions for future research:

(1) In this thesis, we learned the Bayesian network structure from complete data. However, in many real world cases, missing values exist. The issue of incomplete data is not rare in real life data analysis. Bayesian networks have the advantage when facing missing values, due to the probability presentation of the dependence relationship among variables. To deal will missing data at random, one standard approach is the Expectation Maximization (EM) algorithm. Learning Bayesian networks with incomplete data will be one of my future works.
(2) The dataset in this thesis comprises 33 variables. In real life dataset, variables can be thousands levels or more. How to reduct the candidate network space, how to work on the larger and more complex dataset more effectively is our next research direction.

List of Figures

LIST OF FIGURES

List of Tables

LIST OF TABLES

References

[1] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009. (1), (8)

[2] Jie Cheng, David Bell, and Weiru Liu. **Learning bayesian networks from data: An efficient approach based on information theory**, 1997. (1), (7)

[3] Dimitris Margaritis. *Learning Bayesian Network Model Structure from Data*. PhD thesis, Carnegie Mellon University, 2003. (2), (16), (17), (20), (21)

[4] Xiaotong Lin. *Bayesian Network Learning and Applications in Bioinformatics*. PhD thesis, University of Kansas, 2012. (2), (15), (16), (21)

[5] Richard E. Neapolitan. *Learning Bayesian Networks*. Pearson, 2003. (13)

[6] Marco Scutari. *Measures of Variability for Graphical Models*. PhD thesis, University of Padova, 2011. (15), (20), (25)

[7] Marco Scutari. **Bayesian Network Constraint-Based Structure Learning Algorithms: Parallel and Optimized Implementations in the bnlearn R Package**. *Journal of Statistical Software*, **77**, 2017. (16), (20)

[8] Alexandra M. Carvalho. **Scoring functions for learning Bayesian networks**, 2009. (16), (17), (18), (19)

[9] G. F. Cooper and E. Herskovits. **A Bayesian method for the induction of probabilistic networks from data**. *Machine Learning*, **9**:309–347, 1992. (17)

[10] D. Heckerman, D. Geiger, and D. M. Chickering. **Learning Bayesian networks: The combination of knowledge and statistical data**. *Machine Learning*, **20**:197–243, 1995. (17)

[11] David Maxwell Chickering. **Learning Equivalence Classes of Bayesian Network Structures**. *Journal of Machine Learning Research*, **2**:445–498, 2002. (18)

[12] Remco Ronaldus Bouckaert. *Bayesian Belief Networks: from Construction to Inference*. PhD thesis, Utrecht University, 1995. (19), (20)

[13] Hui Liu and Shuigeng Zhou. **A new hybrid method for learning bayesian networks: Separation and reunion**. *Knowledge-Based Systems*, **121**:185–197, 2017. (21)

[14] Moninder Singh and Marco Valtorta. **Construction of Bayesian Network Structures From Data: A Brief Survey and an Efficient Algorithm**. *International Journal of Approximate Reasoning*, **12**:111–131, 1995. (21)

[15] Denver Dash and Marek J. Druzdzel. **A Hybrid Anytime Algorithm for the Construction of Causal Models From Sparse Data**. In KB. Laskey and H. Prade, editors, *In Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 142–149, 1999. (21)

[16] Silvia Acid and Luis M.de Campos. **A hybrid methodology for learning belief networks: BENEDICT**. *International Journal of Approximate Reasoning*, **27**:235–262, 2001. (21)

[17] Ioannis Tsamardinos, Laura E. Brown, and Constantin F. Aliferis. **The max-min hill-climbing Bayesian network structure learning algorithm**. *Machine Learning*, **65**:31–78, 2006. (21)

[18] James W. Myers, Kathryn B. Laskey, and Kenneth A. DeJong. **Learning Bayesian Networks from Incomplete Data using Evolutionary Algorithms**. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation*, **1**, pages 458–465, 1999. (23)

[19] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012. (23)

[20] Marco Scutari. *Bayesian Network Structure Learning, Parameter Learning and Inference*, 1 2018. (24), (29)

[21] Marco Scutarit. **Learning Bayesian Networks with the bnlearn R Package**. *Journal of Statistical Software*, **35**, 2010. (24)

[22] Xavier Robini. *Display and Analyze ROC Curves*, 5 2018. (25)

[23] National Research Council and Institute of Medicine. *Reducing Underage Drinking: A Collective Responsibility*. The National Academies Press, 2004. (25)

[24] Thoroddur Bjarnason, Barbro Andersson, Marie Choquet, Zsuzsanna Elekes, Mark Morgan, and Gertrude Rapinett. **Alcohol culture, family structure and adolescent alcohol use: multilevel modeling of frequency of heavy drinking among 15-16 year old students in 11 European countries.** *Journal of Studies on Alcohol*, **64**:200–208, 2003. (25)

[25] Tom Fawcett. **An introduction to ROC analysis**. *Pattern Recognition Letters*, **27**:861–874, 2006. (29), (34)

[26] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Pearson Education, 2006. (34), (35)

[27] Foster Provost and Tom Fawcett. *Data Science for Business, What You Need to Know about Data Mining and Data-Analytic Thinking*. O'Reilly Media, 2013. (35)

[28] Charles E.Metz. **Basic principles of ROC analysis**. *Seminars in Nuclear Medicine*, **8**:283–298, 1978. (35)

Appendix A

R code

```
library(bnlearn)
library(Rgraphviz)
library(plyr)
library(pROC)
stu_data <- read.csv("~/Desktop/Data Mining folder/student-por.csv")
Alcohol <- (stu_data$Dalc*5+stu_data$Walc*2)/7
stu_data <- data.frame(stu_data, Alcohol)
names(stu_data)
[1] "school"    "sex"       "age"        "address"  "famsize"
[6] "Pstatus"   "Medu"      "Fedu"       "Mjob"     "Fjob"
[11] "reason"   "guardian"  "traveltime" "studytime"
[15] "failures" "schoolsup" "famsup"     "paid"     "activities"
[20] "nursery"  "higher"    "internet"   "romantic" "famrel"
[25] "freetime" "goout"     "Dalc"       "Walc"     "health"
[30] "absences" "G1"        "G2"         "G3"       "Alcohol"
range(stu_data$Alcohol)
# [1] 1 5
count(stu_data$Alcohol >= 3)
#       x freq
# 1 FALSE  574
# 2  TRUE   75
75/(75+574)
# [1] 0.1155624
```

```
count(stu_data$Alcohol > 2)
#       x freq
# 1 FALSE  483
# 2  TRUE  166
166/649
# [1] 0.2557781
count(stu_data$Alcohol >= 2)
#       x freq
# 1 FALSE  449
# 2  TRUE  200
200/(449+200)
# [1] 0.3081664  ## choose 2 to be the threshold, >= 2 is high consumption
# Create a categorical variable namely High based on Alcohol
High <- ifelse(stu_data$Alcohol >= 2, "Yes", "No")
# Appends High to stu_data dataset
stu_data <- data.frame(stu_data, High)
dim(stu_data)
[1] 649  35
# Delete attributes "Dalc" and "Walc"
stu_data <- stu_data[-27]
stu_data <- stu_data[-27]
# Delete attribute "Alcohol"
stu_data <- stu_data[-32]
names(stu_data)
# [1]  "school"     "sex"        "age"       "address"   "famsize"  "Pstatus"
# [7]  "Medu"       "Fedu"       "Mjob"      "Fjob"      "reason"   "guardian"
# [13] "traveltime" "studytime" "failures" "schoolsup" "famsup"   "paid"
# [19] "activities" "nursery"   "higher"   "internet"  "romantic" "famrel"
# [25] "freetime"   "goout"     "health"   "absences"
# [29] "G1"         "G2"        "G3"       "High"
range(stu_data$age)
# [1] 15 22
count(stu_data$age >= 18)
#      x freq
# 1 FALSE  468
# 2  TRUE  181
181/649
# [1] 0.2788906
AGE <- count(stu_data$age)
AGE[order(AGE$freq), ]
plot(AGE[order(AGE$freq), ])
```

```
# separate by age, whether adult or not
stu_data$age[stu_data$age >= 18] <- "Adult"
stu_data$age[stu_data$age <  18] <- "Teenager"
grade3 <- count(stu_data$G3)
gg3 <- grade3[order(grade3$freq), ]
plot(gg3)
range(stu_data$G3)
# [1]  0 19
count(stu_data$G3 > 13)    ## grade=14~19
#       x freq
# 1 FALSE  455
# 2  TRUE  194
194/(194+455)
# [1] 0.2989214
count(stu_data$G3 > 14)    ## grade=15~19 High 20%
#       x freq
# 1 FALSE  518
# 2  TRUE  131
131/649
# [1] 0.201849
count(stu_data$G3 > 15)
#       x freq
# 1 FALSE  567
# 2  TRUE   82
82/649
# [1] 0.1263482
count(stu_data$G3 < 10)    ## grade=0~9   low 15.4%
#       x freq
# 1 FALSE  549
# 2  TRUE  100
100/649
# [1] 0.1540832
1-0.201849-0.1540832     ## grade=10~14  medium 64.4%
# [1] 0.6440678
stu_data$G3_Temp <- NA
stu_data$G3_Temp[stu_data$G3 > 14] <- "Good"
stu_data$G3_Temp[stu_data$G3 < 10] <- "Low"
stu_data$G3_Temp[stu_data$G3 >  9 & stu_data$G3<15] <- "Medium"
grade2 <- count(stu_data$G2)
gg2 <- grade2[order(grade2$freq), ]
```

```
plot(gg2)
count(stu_data$G2 < 10)
#    x freq
# 1 FALSE  504
# 2  TRUE  145
145/649
# [1] 0.2234206
count(stu_data$G2 > 14)
#       x freq
# 1 FALSE  551
# 2  TRUE   98
98/649
# [1] 0.1510015
count(stu_data$G2 < 9)          ## 0~8 Low 11.2%
#       x freq
# 1 FALSE  576
# 2  TRUE   73
73/649
# [1] 0.1124807
count(stu_data$G2 > 13)         ## 14~19 Good 23.4%
#       x freq
# 1 FALSE  497
# 2  TRUE  152
152/649
# [1] 0.2342065
stu_data$G2_Temp <- NA
stu_data$G2_Temp[stu_data$G2 > 13] <- "Good"
stu_data$G2_Temp[stu_data$G2 < 9] <- "Low"
stu_data$G2_Temp[stu_data$G2 > 8 & stu_data$G2 < 14] <- "Medium"
grade1 <- count(stu_data$G1)
gg1 <- grade1[order(grade1$freq), ]
plot(gg1)
count(stu_data$G1 < 10)
#       x freq
# 1 FALSE  492
# 2  TRUE  157
157/649
# [1] 0.2419106
count(stu_data$G1 < 9)        ### G1 14.1% Low
#       x freq
# 1 FALSE  557
# 2  TRUE   92
92/649
# [1] 0.1417565
```

```
count(stu_data$G1 > 14)
#       x freq
# 1 FALSE  568
# 2  TRUE   81
81/649
# [1] 0.1248074
count(stu_data$G1 > 13)       ### G1 23% Good
#       x freq
# 1 FALSE  497
# 2  TRUE  152
152/649
# [1] 0.2342065

stu_data$G1_Temp <- NA
stu_data$G1_Temp[stu_data$G1 > 13] <- "Good"
stu_data$G1_Temp[stu_data$G1 < 9] <- "Low"
stu_data$G1_Temp[stu_data$G1 > 8 & stu_data$G1 < 14] <- "Medium"
stu_data <- stu_data[-29]  # delete G1
stu_data <- stu_data[-29]  # delete G2
stu_data <- stu_data[-29]  # delete G3

set.seed(666)
test <- sample(1:nrow(stu_data), nrow(stu_data)/3)
train <- -test
training <- stu_data[train,]
training_data <- as.data.frame(lapply(training, as.factor))
testing <- stu_data[test,]
testing_data <- as.data.frame(lapply(testing, as.factor))


bl <- matrix(c("studytime", "sex",                ### blacklist
               "High"     , "sex",
               "romantic" , "sex",
               "romantic" , "age",
               "G1_Temp"  , "age",
               "G1_Temp"  , "sex",
               "G2_Temp"  , "age",
               "higher"   , "age",
               "failures" , "age" ), ncol=2, byrow=TRUE)
```

```
wl <- matrix(c("guardian", "High",                ### whitelist
               "famrel"  , "guardian",
               "age"     , "higher",
               "Medu"    , "Mjob",
               "Fedu"    , "Fjob",
               "Fjob"    , "famrel",
               "Mjob"    , "famrel",
               "goout"   , "absences",
               "age"     , "guardian",
               "age"     , "goout",
               "age"     , "High",
               "sex"     , "guardian",
               "sex"     , "High",
               "goout"   , "High",
               "Medu"    , "famrel",
               "Fedu"    , "famrel",
               "absences", "failures",
               "High"    , "G3_Temp",
               "High"    , "failures",
               "High"    , "absences"), ncol=2, byrow=TRUE)

### #### #### #### #### #### ### K2 score ### #### #### #### #### #### #### #### ##
hck21 <- hc(training_data, score="k2")
hck22 <- hc(training_data, score="k2", restart=20)
hck23 <- hc(training_data, score="k2",blacklist=bl, whitelist=wl,restart=20)
tabuk21 <- tabu(training_data, tabu=20,optimized=TRUE, score="k2")
tabuk22 <- tabu(training_data, blacklist=bl, whitelist=wl, tabu=20,
                                              optimized=TRUE, score="k2")
pr.hck21 <- predict(hck21,"High",testing_data,method="bayes-lw", prob=TRUE)
pr.hck22 <- predict(hck22,"High",testing_data,method="bayes-lw", prob=TRUE)
pr.hck23 <- predict(hck23,"High",testing_data,method="bayes-lw", prob=TRUE)
pr.tabuk21 <- predict(tabuk21,"High",testing_data,method="bayes-lw", prob=TRUE)
pr.tabuk22 <- predict(tabuk22,"High",testing_data,method="bayes-lw", prob=TRUE)
mean(pr.hck21!=testing_data$High)   # error rates
# [1] 0.2824074
mean(pr.hck22!=testing_data$High)
# [1] 0.2731481
mean(pr.hck23!=testing_data$High)
# [1] 0.1296296
mean(pr.tabuk21!=testing_data$High)
# [1] 0.2962963
mean(pr.tabuk22!=testing_data$High)
# [1] 0.1157407
```

```
r_hck21 <- attributes(pr.hck21)$prob[1,]
r_hck22 <- attributes(pr.hck22)$prob[1,]
r_hck23 <- attributes(pr.hck23)$prob[1,]
r_tabuk21 <- attributes(pr.tabuk21)$prob[1,]
r_tabuk22 <- attributes(pr.tabuk22)$prob[1,]
roc_hck21 <- roc(testing_data$High, r_hck21)
roc_hck22 <- roc(testing_data$High, r_hck22)
roc_hck23 <- roc(testing_data$High, r_hck23)
roc_tabuk21 <- roc(testing_data$High, r_tabuk21)
roc_tabuk22 <- roc(testing_data$High, r_tabuk22)

pdf(file="k2.pdf", width=8, height=12)
 par(mfrow=c(3,2))
 graphviz.plot(hck21, main="K2 Hill-climbing", sub="A")
 graphviz.plot(hck22, main="K2 Hill-climbing, 20 restarts", sub="B")
 graphviz.plot(hck23, main="K2 Hill-climbing with BL and WL, 20 restarts", sub="C")
 graphviz.plot(tabuk21, main="K2 Tabu", sub="D")
 graphviz.plot(tabuk22, main="K2 Tabu with BL and WL", sub="E")
 plot(roc_hck21, col="red")
 lines(roc_hck22, col="green")
 lines(roc_hck23, col="orange")
 lines(roc_tabuk21, col="blue")
 lines(roc_tabuk22, col="black")
 legend("bottomright", legend=c("A", "B", "C", "D", "E"), lty=1,
                       col=c("red", "green", "orange", "blue", "black"), lwd=1)
dev.off()

mean(pr.hck21!=testing_data$High)   # error rates
# [1] 0.2824074
mean(pr.hck22!=testing_data$High)
# [1] 0.2731481
mean(pr.hck23!=testing_data$High)
# [1] 0.1296296
mean(pr.tabuk21!=testing_data$High)
# [1] 0.2962963
mean(pr.tabuk22!=testing_data$High)
# [1] 0.1157407
r_hck21 <- attributes(pr.hck21)$prob[1,]
r_hck22 <- attributes(pr.hck22)$prob[1,]
r_hck23 <- attributes(pr.hck23)$prob[1,]
r_tabuk21 <- attributes(pr.tabuk21)$prob[1,]
r_tabuk22 <- attributes(pr.tabuk22)$prob[1,]
```

```
roc_hck21 <- roc(testing_data$High, r_hck21)
roc_hck22 <- roc(testing_data$High, r_hck22)
roc_hck23 <- roc(testing_data$High, r_hck23)
roc_tabuk21 <- roc(testing_data$High, r_tabuk21)
roc_tabuk22 <- roc(testing_data$High, r_tabuk22)

pdf(file="k2.pdf", width=8, height=12)
 par(mfrow=c(3,2))
 graphviz.plot(hck21, main="K2 Hill-climbing", sub="A")
 graphviz.plot(hck22, main="K2 Hill-climbing, 20 restarts", sub="B")
 graphviz.plot(hck23, main="K2 Hill-climbing with BL and WL, 20 restarts", sub="C")
 graphviz.plot(tabuk21, main="K2 Tabu", sub="D")
 graphviz.plot(tabuk22, main="K2 Tabu with BL and WL", sub="E")
 plot(roc_hck21, col="red")
 lines(roc_hck22, col="green")
 lines(roc_hck23, col="orange")
 lines(roc_tabuk21, col="blue")
 lines(roc_tabuk22, col="black")
 legend("bottomright", legend=c("A", "B", "C", "D", "E"),  lty=1,
                       col=c("red", "green", "orange", "blue", "black"), lwd=1)
dev.off()

roc_hck21
# roc.default(response = testing_data$High, predictor = r_hck21)
# Data: r_hck21 in 153 controls (testing_data$High 0) > 63 cases (testing_data$High 1).
# Area under the curve: 0.6797
roc_hck22
# 0.7052
roc_hck23
# 0.9101
roc_tabuk21
# 0.7606
roc_tabuk22
# 0.9052
### #### #### #### #### #### ## BIC score ### #### #### #### #### #### #
set.seed(600)
test <- sample(1:nrow(stu_data), nrow(stu_data)/3)
train <- -test
training <- stu_data[train,]
training_data <- as.data.frame(lapply(training, as.factor))
testing <- stu_data[test,]
testing_data <- as.data.frame(lapply(testing, as.factor))
hcbic1 <- hc(training_data, score="bic")
hcbic2 <- hc(training_data, score="bic", restart=20)
hcbic3 <- hc(training_data, score="bic",blacklist=bl, whitelist=wl,restart=20)
```

```
tabubic1 <- tabu(training_data, tabu=20,optimized=TRUE, score="bic")
tabubic2 <- tabu(training_data, blacklist=bl, whitelist=wl, tabu=20,
               optimized=TRUE, score="bic")
pr.hcbic1 <- predict(hcbic1,"High",testing_data,method="bayes-lw", prob=TRUE)
pr.hcbic2 <- predict(hcbic2,"High",testing_data,method="bayes-lw", prob=TRUE)
pr.hcbic3 <- predict(hcbic3,"High",testing_data,method="bayes-lw", prob=TRUE)
pr.tabubic1 <- predict(tabubic1,"High",testing_data,method="bayes-lw", prob=TRUE)
pr.tabubic2 <- predict(tabubic2,"High",testing_data,method="bayes-lw", prob=TRUE)
mean(pr.hcbic1 != testing_data$High)
# [1] 0.2731481
mean(pr.hcbic2 != testing_data$High)
# [1] 0.3055556
mean(pr.hcbic3 != testing_data$High)
# [1] 0.1018519
mean(pr.tabubic1 != testing_data$High)
# [1] 0.2824074
mean(pr.tabubic2 != testing_data$High)
# [1] 0.09722222
r_hcbic1 <- attributes(pr.hcbic1)$prob[1,]
r_hcbic2 <- attributes(pr.hcbic2)$prob[1,]
r_hcbic3 <- attributes(pr.hcbic3)$prob[1,]
r_tabubic1 <- attributes(pr.tabubic1)$prob[1,]
r_tabubic2 <- attributes(pr.tabubic2)$prob[1,]
roc_hcbic1 <- roc(testing_data$High, r_hcbic1)
roc_hcbic2 <- roc(testing_data$High, r_hcbic2)
roc_hcbic3 <- roc(testing_data$High, r_hcbic3)
roc_tabubic1 <- roc(testing_data$High, r_tabubic1)
roc_tabubic2 <- roc(testing_data$High, r_tabubic2)
pdf(file="bic.pdf", width=8, height=12)
 par(mfrow=c(3,2))
 graphviz.plot(hcbic1, main="BIC Hill-climbing", sub="A")
 graphviz.plot(hcbic2, main="BIC Hill-climbing, 20 restarts", sub="B")
 graphviz.plot(hcbic3, main="BIC Hill-climbing with BL and WL, 20 restarts", sub="C")
 graphviz.plot(tabubic1, main="BIC Tabu", sub="D")
 graphviz.plot(tabubic2, main="BIC Tabu with BL and WL", sub="E")
 plot(roc_hcbic1, col="red")
 lines(roc_hcbic2, col="green")
 lines(roc_hcbic3, col="orange")
 lines(roc_tabubic1, col="blue")
 lines(roc_tabubic2, col="black")
 legend("bottomright", legend=c("A", "B", "C", "D", "E"),  lty=1,
                       col=c("red", "green", "orange", "blue", "black"), lwd=1)
dev.off()
```

```
roc_hcbic1
# Call:
# roc.default(response = testing_data$High, predictor = r_hcbic1)
# Data: r_hcbic1 in 147 controls (testing_data$High 0) > 69 cases (testing_data$High 1).
# Area under the curve: 0.6272
roc_hcbic2
# Call:
# roc.default(response = testing_data$High, predictor = r_hcbic2)
# Data: r_hcbic2 in 147 controls (testing_data$High 0) > 69 cases (testing_data$High 1).
# Area under the curve: 0.6582
roc_hcbic3
# Call:
# roc.default(response = testing_data$High, predictor = r_hcbic3)
# Data: r_hcbic3 in 147 controls (testing_data$High 0) > 69 cases (testing_data$High 1).
# Area under the curve: 0.9232
roc_tabubic1
# Call:
# roc.default(response = testing_data$High, predictor = r_tabubic1)
# Data: r_tabubic1 in 147 controls (testing_data$High 0) > 69 cases (testing_data$High 1).
# Area under the curve: 0.705
roc_tabubic2
# Call:
# roc.default(response = testing_data$High, predictor = r_tabubic2)
# Data: r_tabubic2 in 147 controls (testing_data$High 0) > 69 cases (testing_data$High 1).
# Area under the curve: 0.925

### #### #### #### #### ## AIC score ### #### #### #### #### #
set.seed(111)
test <- sample(1:nrow(stu_data), nrow(stu_data)/3)
train <- -test
training <- stu_data[train,]
training_data <- as.data.frame(lapply(training, as.factor))
testing <- stu_data[test,]
testing_data <- as.data.frame(lapply(testing, as.factor))
hcaic1 <- hc(training_data, score="aic")
hcaic2 <- hc(training_data, score="aic", restart=20)
hcaic3 <- hc(training_data, score="aic",blacklist=bl, whitelist=wl,restart=20)
tabuaic1 <- tabu(training_data, tabu=20,optimized=TRUE, score="aic")
tabuaic2 <- tabu(training_data, blacklist=bl, whitelist=wl, tabu=20,
                                                optimized=TRUE, score="aic")
pr.hcaic1 <- predict(hcaic1,"High",testing_data,method="bayes-lw", prob=TRUE)
pr.hcaic2 <- predict(hcaic2,"High",testing_data,method="bayes-lw", prob=TRUE)
pr.hcaic3 <- predict(hcaic3,"High",testing_data,method="bayes-lw", prob=TRUE)
pr.tabuaic1 <- predict(tabuaic1,"High",testing_data,method="bayes-lw", prob=TRUE)
pr.tabuaic2 <- predict(tabuaic2,"High",testing_data,method="bayes-lw", prob=TRUE)
```

```
mean(pr.hcaic1 != testing_data$High)
# [1] 0.1990741
mean(pr.hcaic2 != testing_data$High)
# [1] 0.1944444
mean(pr.hcaic3 != testing_data$High)
# [1] 0.08333333
mean(pr.tabuaic1 != testing_data$High)
# [1] 0.1851852
mean(pr.tabuaic2 != testing_data$High)
# [1] 0.08333333
r_hcaic1 <- attributes(pr.hcaic1)$prob[1,]
r_hcaic2 <- attributes(pr.hcaic2)$prob[1,]
r_hcaic3 <- attributes(pr.hcaic3)$prob[1,]
r_tabuaic1 <- attributes(pr.tabuaic1)$prob[1,]
r_tabuaic2 <- attributes(pr.tabuaic2)$prob[1,]
roc_hcaic1 <- roc(testing_data$High, r_hcaic1)
roc_hcaic2 <- roc(testing_data$High, r_hcaic2)
roc_hcaic3 <- roc(testing_data$High, r_hcaic3)
roc_tabuaic1 <- roc(testing_data$High, r_tabuaic1)
roc_tabuaic2 <- roc(testing_data$High, r_tabuaic2)
pdf(file="aic.pdf", width=8, height=12)
 par(mfrow=c(3,2))
 graphviz.plot(hcaic1, main="AIC Hill-climbing", sub="A")
 graphviz.plot(hcaic2, main="AIC Hill-climbing, 20 restarts", sub="B")
 graphviz.plot(hcaic3, main="AIC Hill-climbing with BL and WL, 20 restarts", sub="C")
 graphviz.plot(tabuaic1, main="AIC Tabu", sub="D")
 graphviz.plot(tabuaic2, main="AIC Tabu with BL and WL", sub="E")
 plot(roc_hcaic1, col="red")
 lines(roc_hcaic2, col="green")
 lines(roc_hcaic3, col="orange")
 lines(roc_tabuaic1, col="blue")
 lines(roc_tabuaic2, col="black")
 legend("bottomright", legend=c("A", "B", "C", "D", "E"),  lty=1,
                       col=c("red", "green", "orange", "blue", "black"), lwd=1)
dev.off()
roc_hcaic1
# Call:
# roc.default(response = testing_data$High, predictor = r_hcaic1)
# Data: r_hcaic1 in 152 controls (testing_data$High 0) > 64 cases (testing_data$High 1).
# Area under the curve: 0.7478
roc_hcaic2
# Call:
# roc.default(response = testing_data$High, predictor = r_hcaic2)
# Data: r_hcaic2 in 152 controls (testing_data$High 0) > 64 cases (testing_data$High 1).
# Area under the curve: 0.7487
```

```
roc_hcaic3
# Call:
# roc.default(response = testing_data$High, predictor = r_hcaic3)
# Data: r_hcaic3 in 152 controls (testing_data$High 0) > 64 cases (testing_data$High 1).
# Area under the curve: 0.8899
roc_tabuaic1
# Call:
# roc.default(response = testing_data$High, predictor = r_tabuaic1)
# Data: r_tabuaic1 in 152 controls (testing_data$High 0) > 64 cases (testing_data$High 1).
# Area under the curve: 0.7506
roc_tabuaic2
# Call:
# roc.default(response = testing_data$High, predictor = r_tabuaic2)
# Data: r_tabuaic2 in 152 controls (testing_data$High 0) > 64 cases (testing_data$High 1).
# Area under the curve: 0.8917

### #### #### #### #### ### BDe score ### #### #### #### ###
set.seed(222)
test <- sample(1:nrow(stu_data), nrow(stu_data)/3)
train <- -test
training <- stu_data[train,]
training_data <- as.data.frame(lapply(training, as.factor))
testing <- stu_data[test,]
testing_data <- as.data.frame(lapply(testing, as.factor))
hcbde1 <- hc(training_data, score="bde")
hcbde2 <- hc(training_data, score="bde", restart=20)
hcbde3 <- hc(training_data, score="bde",blacklist=bl, whitelist=wl,restart=20)
tabubde1 <- tabu(training_data, tabu=20,optimized=TRUE, score="bde")
tabubde2 <- tabu(training_data, blacklist=bl, whitelist=wl, tabu=20,
                                                optimized=TRUE, score="bde")
pr.hcbde1 <- predict(hcbde1,"High",testing_data,method="bayes-lw", prob=TRUE)
pr.hcbde2 <- predict(hcbde2,"High",testing_data,method="bayes-lw", prob=TRUE)
pr.hcbde3 <- predict(hcbde3,"High",testing_data,method="bayes-lw", prob=TRUE)
pr.tabubde1 <- predict(tabubde1,"High",testing_data,method="bayes-lw", prob=TRUE)
pr.tabubde2 <- predict(tabubde2,"High",testing_data,method="bayes-lw", prob=TRUE)
mean(pr.hcbde1 != testing_data$High)
# [1] 0.2638889
mean(pr.hcbde2 != testing_data$High)
# [1] 0.2638889
mean(pr.hcbde3 != testing_data$High)
# [1] 0.1157407
mean(pr.tabubde1 != testing_data$High)
# [1] 0.2314815
mean(pr.tabubde2 != testing_data$High)
# [1] 0.1157407
```

```
r_hcbde1 <- attributes(pr.hcbde1)$prob[1,]
r_hcbde2 <- attributes(pr.hcbde2)$prob[1,]
r_hcbde3 <- attributes(pr.hcbde3)$prob[1,]
r_tabubde1 <- attributes(pr.tabubde1)$prob[1,]
r_tabubde2 <- attributes(pr.tabubde2)$prob[1,]
roc_hcbde1 <- roc(testing_data$High, r_hcbde1)
roc_hcbde2 <- roc(testing_data$High, r_hcbde2)
roc_hcbde3 <- roc(testing_data$High, r_hcbde3)
roc_tabubde1 <- roc(testing_data$High, r_tabubde1)
roc_tabubde2 <- roc(testing_data$High, r_tabubde2)

pdf(file="bde.pdf", width=8, height=12)
 par(mfrow=c(3,2))
 graphviz.plot(hcbde1, main="BDe Hill-climbing", sub="A")
 graphviz.plot(hcbde2, main="BDe Hill-climbing, 20 restarts", sub="B")
 graphviz.plot(hcbde3, main="BDe Hill-climbing with BL and WL, 20 restarts", sub="C")
 graphviz.plot(tabubde1, main="BDe Tabu", sub="D")
 graphviz.plot(tabubde2, main="BDe Tabu with BL and WL", sub="E")
 plot(roc_hcbde1, col="red")
 lines(roc_hcbde2, col="green")
 lines(roc_hcbde3, col="orange")
 lines(roc_tabubde1, col="blue")
 lines(roc_tabubde2, col="black")
 legend("bottomright", legend=c("A", "B", "C", "D", "E"),  lty=1,
                       col=c("red", "green", "orange", "blue", "black"), lwd=1)
dev.off()

roc_hcbde1
# Call:
# roc.default(response = testing_data$High, predictor = r_hcbde1)
# Data: r_hcbde1 in 156 controls (testing_data$High 0) > 60 cases (testing_data$High 1).
# Area under the curve: 0.72
roc_hcbde2
# Call:
# roc.default(response = testing_data$High, predictor = r_hcbde2)
# Data: r_hcbde2 in 156 controls (testing_data$High 0) > 60 cases (testing_data$High 1).
# Area under the curve: 0.7324
roc_hcbde3
# Call:
# roc.default(response = testing_data$High, predictor = r_hcbde3)
# Data: r_hcbde3 in 156 controls (testing_data$High 0) > 60 cases (testing_data$High 1).
# Area under the curve: 0.8927
```

```
roc_tabubde1
# Call:
# roc.default(response = testing_data$High, predictor = r_tabubde1)
# Data: r_tabubde1 in 156 controls (testing_data$High 0) > 60 cases (testing_data$High 1).
# Area under the curve: 0.7435
roc_tabubde2
# Call:
# roc.default(response = testing_data$High, predictor = r_tabubde2)
# Data: r_tabubde2 in 156 controls (testing_data$High 0) > 60 cases (testing_data$High 1).
# Area under the curve: 0.8962

### #### #### #### #### ## compare 4 scores### #### #### #### #### #### ###
plot(roc_hck23, col="red", main="ROC analysis with Blacklist and Whitelist")
lines(roc_tabubic2, col="green")
lines(roc_tabuaic2, col="orange")
lines(roc_tabubde2, col="blue")
legend("bottomright", legend=c("hc K2", "tabu BIC", "tabu AIC", "tabu BDe"),  lty=1,
                      col=c("red", "green", "orange", "blue"), lwd=1)

roc_hck23
# Call:
# roc.default(response = testing_data$High, predictor = r_hck23)
# Data: r_hck23 in 153 controls (testing_data$High 0) > 63 cases (testing_data$High 1).
# Area under the curve: 0.9101

roc_tabubic2
# Call:
# roc.default(response = testing_data$High, predictor = r_tabubic2)
# Data: r_tabubic2 in 147 controls (testing_data$High 0) > 69 cases (testing_data$High 1).
# Area under the curve: 0.925

roc_tabuaic2
# Call:
# roc.default(response = testing_data$High, predictor = r_tabuaic2)
# Data: r_tabuaic2 in 152 controls (testing_data$High 0) > 64 cases (testing_data$High 1).
# Area under the curve: 0.8917

roc_tabubde2
# Call:
# roc.default(response = testing_data$High, predictor = r_tabubde2)
# Data: r_tabubde2 in 156 controls (testing_data$High 0) > 60 cases (testing_data$High 1).
# Area under the curve: 0.8962
```

STATEMENT

Name: Cui Hao

Neptun code: OMYV1Z

ELTE Faculty of Science, Field of studies: Mathematics

Title of thesis: Learning Bayesian Network Structure from Data

As the author of this thesis I declare that it is my own work and that I have acknowledged and referenced the work and ideas of others.

May 31st, 2018....., **Budapest**

崔浩 Cui Hao

signature