Scoring functions for learning Bayesian networks

INESC-ID Tec. Rep. 54/2009

Apr 2009

Alexandra M. Carvalho
IST, TULisbon/INESC-ID
asmc@inesc-id.pt

Abstract

The aim of this work is to benchmark scoring functions used by Bayesian network learning algorithms in the context of classification. We considered both information-theoretic scores, such as LL, AIC, BIC/MDL, NML and MIT, and Bayesian scores, such as K2, BD, BDe and BDeu. We tested the scores in a classification task by learning the optimal TAN classifier with benchmark datasets. We conclude that, in general, information-theoretic scores perform better than Bayesian scores.

1 Introduction

Bayesian networks [Pea88] allow efficient and accurate representation of the joint probability distribution over a set of random variables. For this reason, they have been widely used in several domains of application where uncertainty plays an important role, like medical diagnosis and modeling DNA binding sites. Learning Bayesian networks consists of finding the network that best fits, for a certain scoring function, the data. This problem is not straightforward. Cooper [Coo90] showed that the inference of a general Bayesian network is a NP-hard problem, and later, Dagum and Luby [DL93] showed that even finding an approximate solution is NP-hard.

These results led the community to search for the largest subclass of Bayesian networks for which there is an efficient structure learning algorithm. First attempts confined the network

to tree structures and used Edmonds [Edm67] and Chow-Liu [CL68] optimal branching algorithms to learn the network. More general classes of Bayesian networks have eluded efforts to develop efficient learning algorithms. Indeed, Chickering [Chi96] showed that learning the structure of a Bayesian network is NP-hard even for networks constrained to have in-degree at most 2. Later, Dasgupta [Das99] showed that even learning 2-polytrees[1] is NP-hard. Due to these hardness results exact polynomial-time bounded approaches for learning Bayesian networks have been restricted to tree structures.

Consequently, the standard methodology for addressing the problem of learning Bayesian networks became heuristic search, based on scoring metrics optimization, conducted over some search space. Many algorithms have been proposed along these lines, varying both on the formulation of the search space (network structures, equivalence classes of network structures and orderings over the network variables), and on the algorithm to search the space (greedy hill-climbing, simulated annealing, genetic algorithms, tabu search, etc). Nevertheless, in all these algorithms the search is guided by a scoring function that evaluates the degree of fitness between the network and the data.

The aim of this work is to study scoring functions used by learning algorithms based on the *score+search* paradigm in the context of classification, namely, for learning the optimal TAN classifier [FGG97]. We also want to empirically evaluate the merits of each score by means of a comparative experimental study over benchmark datasets. Well known scores are those based on information theory, such as *log-likelihood* (LL), *Akaike information criterion* (AIC), *Bayesian information criterion* (BIC) (also improperly called *minimum description length* (MDL)), *normalized maximum likelihood* (NML) and *mutual information test* (MIT), and Bayesian scoring functions such as K2, *Bayesian Dirichlet* (BD) and its variants (BDe and BDeu).

The paper is organized as follows. In Section 2, we briefly revise Bayesian networks and learning algorithms for tree-structured Bayesian networks. Such algorithms are based on the *score+search* paradigm, and so all scoring functions known in the literature are presented, justified, and classified among important properties. In Section 3, we revise Bayesian network classifiers and extend previous algorithms for learning optimal TAN classifiers. In Section 4 we present experimental results which evaluates the merits of each score. Finally, in Section

[1] A *polytree* is a directed acyclic graph such that, given two nodes, there are not two different paths from one to another. A 2-polytree is a polytree where each node has at most in-degree 2.

5 we draw some conclusions and discuss future work.

2 Learning Bayesian networks

A *finite random variable* X over D is a random variable that takes values over a finite set $D \subseteq \mathbb{R}$. A n-dimensional *finite random vector* $\mathbf{X}$ over D is a random vector where each component X_i is a random variable over D for all $i = 1, \ldots, n$. The random vectors $\mathbf{X}$ and $\mathbf{Y}$ are said to be *conditionally independent* given a random vector over $\mathbf{Z}$ if $P(\mathbf{x}|\mathbf{y}, \mathbf{z}) = P(\mathbf{x}|\mathbf{z})$.

Definition 2.1 (Bayesian network) A *n-dimensional Bayesian network* (BN) is a triple $B = (\mathbf{X}, G, \Theta)$ where:

- $\mathbf{X}$ is a n-dimensional finite random vector where each random variable X_i ranged over by a finite domain D_i. Henceforward, we denote the joint domain by $\mathbf{D} = \prod_{i=1}^{n} D_i$.
- $G = (N, E)$ is a directed acyclic graph (DAG) with nodes $N = \{X_1, \ldots, X_n\}$ and edges E representing direct dependencies between the variables.
- Θ encodes the parameters $\{\theta_{ijk}\}_{i \in 1 \ldots n,\, j \in D_{\Pi_{X_i}},\, k \in D_i}$ of the network, where

$$\theta_{ijk} = P_B(X_i = x_{ik} | \Pi_{X_i} = w_{ij}),$$

  Π_{X_i} denotes the set of parents of X_i in G, $D_{\Pi_{X_i}}$ denotes the joint domain of the variables in Π_{X_i}, x_{ik} is the k-th value of X_i and w_{ij} is the j-th configuration of Π_{X_i}.

A Bayesian network defines a unique joint probability distribution over $\mathbf{X}$ given by

$$P_B(X_1, \ldots, X_n) = \prod_{i=1}^{n} P_B(X_i | \Pi_{X_i}). \tag{1}$$

We denote the set of all Bayesian networks with n variables by $\mathcal{B}_n$.

Informally, a Bayesian network encodes the independence assumptions over the component random variables of $\mathbf{X}$. An edge (j, i) in E represents a direct dependency of X_i from X_j. Moreover, X_i is conditionally independent of its non descendants given its parents Π_{X_i} in G. In addition to the graph structure, it is necessary to specify the parameters that quantify the network. This is where the third component of the triple takes place, it specifies the conditional probability distribution θ_{ijk} at each node for each possible value $i \in 1 \ldots n$, $j \in D_{\Pi_{X_i}}$ and $k \in D_i$.

The problem of learning a Bayesian network given data T consists on finding the Bayesian network that best fits the data T. By a Bayesian network that best fits the data T one could be tempted to consider a Bayesian network that maximizes the probability of generating T. However, this approach is not very useful because the learned network overffits. In order to quantify the fitting of a Bayesian network a *scoring function* $\phi : \mathcal{B}_n \times \mathbf{D}^N \to \mathbb{R}$ is considered. The scoring function should be asymptotically correct, that is, the learned distribution with maximum score should converge, with high probability, to the underlying distribution as the size of the data increases. In this context, the problem of learning a Bayesian network can be recasted in the following optimization problem.

Definition 2.2 (Learning a Bayesian network) Given a data $T = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$ and a scoring function ϕ, the *problem of learning a Bayesian network* is to find a Bayesian network $B \in \mathcal{B}_n$ that maximizes the value $\phi(B, T)$.

Learning Bayesian networks is not straightforward. Cooper [Coo90] showed that the inference of a general Bayesian network is a NP-hard problem, and later, Dagum and Luby [DL93] showed that even finding an approximate solution is NP-hard.

2.1 Scoring functions for learning Bayesian networks

Several scoring functions for learning Bayesian networks have been proposed in the literature [dC06, YC02]. It is common to classify scoring functions into two main categories: Bayesian and information-theoretic. In general, for efficiency purposes, these scores need to decompose over the network structure. 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, that is, equivalent networks must score the same.

Next, we survey scoring functions for learning Bayesian networks and classify them according the decomposability and score-equivalence properties. We start by introducing some notation. The number of states of the finite random variable X_i is r_i. The number of possible configurations of the parent set Π_{X_i} of X_i is q_i, that is,

$$q_i = \prod_{X_j \in \Pi_{X_i}} r_j.$$

A configuration of Π_{X_i} is represented by w_{ij} $(1 \leq j \leq q_i)$. N_{ijk} is the number of instances in the data T where the variable X_i takes its k-th value x_{ik} and the variables in Π_{X_i} take their j-th configuration w_{ij}. N_{ij} is the number of instances in the data T where the variables in Π_{X_i} take their j-th configuration w_{ij}, that is,

$$N_{ij} = \sum_{k=1}^{r_i} N_{ijk}.$$

Finally, the total number of instances in the data T is N.

2.1.1 Bayesian scoring functions

The general idea of Bayesian scoring functions is to compute the posterior probability distribution, starting from a prior probability distribution on the possible networks, conditioned to data T, that is, $P(B|T)$. The best network is the one that maximizes the posterior probability. Since the term $P(T)$ is the same for all possible networks, in practice, for comparative purposes, computing $P(B,T)$ is sufficient. Moreover, as it is easier to work in the logarithmic space, the scoring functions use the value $\log(P(B,T))$ instead of $P(B,T)$.

BD scoring function

Heckerman et al. [HGC95] proposed the *Bayesian Dirichlet* (BD) score by making four assumptions on $P(B,T)$. The first one assumes that data T is *exchangeable*, that is, if an instance of the data is exchanged with another instance, the exchanged data has the same probability as the original one. De Finetti [dF37] showed that exchangeable instances can only be explained by multinomial samples, which lead to formalizing the first assumption as a multinomial sample hypothesis. Let us now introduce the needed notation.

We define,

$$\Theta_G = \{\Theta_i\}_{i=1,\ldots,n},$$

$$\Theta_i = \{\Theta_{ij}\}_{j=1,\ldots,q_i},$$

$$\Theta_{ij} = \{\theta_{ijk}\}_{k=1,\ldots,r_i}.$$

That is, Θ_G encodes the parameters of a Bayesian network B with underlying DAG G, Θ_i encodes the parameters concerning only the variable X_i of $\mathbf{X}$ in B, and Θ_{ij} encodes the parameters for variable X_i given that its parents take their j-th configuration.

Assumption 1 (Multinomial sample) For any data $T = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$, Bayesian network B, variable X_i of $\mathbf{X}$ in B and instance $\mathbf{y}_t \in T$,

$$P_B(\mathbf{y}_{ti} = x_{ik} | \mathbf{y}_{t\Pi_{X_i}} = w_{ij}, T_t) = P_B(X_i = x_{ik} | \Pi_{X_i} = w_{ij}) = \theta_{ijk}$$

for $k = 1, \ldots, r_i$ and $j = 1, \ldots, q_i$, where $T_t = \{\mathbf{y}_1, \ldots, \mathbf{y}_{t-1}\}$.

The multinomial sample assumption asserts that the probability of observing the t-th instance of data is conditionally independent on the previous observations T_t given B, like it is usual for the multinomial distribution.

The second assumption assumes that parameters Θ_{ij} have a Dirichlet distribution. This hypothesis is convenient because the Dirichlet distribution is *closed under multinomial sampling*, that is, if the prior distribution is Dirichlet, the posterior distribution, given a multinomial sample, is also Dirichlet.

Assumption 2 (Dirichlet) Given a directed acyclic graph G such that $P(G) > 0$ then Θ_{ij} is Dirichlet for all Θ_{ij} in Θ_G.

The Dirichlet assumption imposes that the probability density function for Θ_{ij} is given by

$$\rho(\Theta_{ij}|G) = c \prod_{k=1}^{r_i} \theta_{ijk}^{N'_{ijk}-1} \tag{2}$$

with $N'_{ijk} > 0$, where $\{N'_{ijk}\}_{k=1\ldots r_i}$ are the hyperparameters (exponents) of the Dirichlet distribution. Moreover, when Θ_{ij} is Dirichlet, the expectation of θ_{ijk} is given by

$$E(\theta_{ijk}) = \frac{N'_{ijk}}{N'_{ij}} \tag{3}$$

where

$$N'_{ij} = \sum_{k=1}^{r_i} N'_{ijk}.$$

At the light of Equations (2) and (3), the hyperparameter N'_{ijk} can be interpreted as $N'_{ijk} - 1$ pseudo-counts (prior to data observation) for the k-th value of X_i given that its parents were in their j-th configuration. If the prior distribution of Θ_G fulfills Equation (2), then the posteriori probability of Θ_G given a multinomial sampled data T is

$$\rho(\Theta_{ij}|T, G) = c \prod_{k=1}^{r_i} \theta_{ijk}^{N'_{ijk}-1+N_{ijk}}.$$

It is worthwhile noticing that the Dirichlet assumption will be useful in practice to smooth the parameters learned from small data.

The remaining two assumptions have in mind simplifying the computation of $P(B,T)$. The third hypothesis imposes that the parameters associated with each variable in the network are independent, and, moreover, the parameters associated with each instance of the parents of a variable are also independent.

Assumption 3 (Parameter independence) Given a directed acyclic graph G such that $P(G) > 0$ then

1. $\rho(\Theta_G|G) = \prod_{i=1}^{n} \rho(\Theta_i|G)$ (global parameter independence), and
2. $\rho(\Theta_i|G) = \prod_{j=1}^{q_i} \rho(\Theta_{ij}|G)$ for all $i = 1, \ldots, n$ (local parameter independence).

Finally, the fourth assumption states that the density for the parameters Θ_{ij} depends only on X_i and its parents, that is, on the local structure of X_i.

Assumption 4 (Parameter modularity) Given two directed acyclic graphs, G and G', such that $P(G) > 0$ and $P(G') > 0$, if X_i has the same parents in G and G', then

$$\rho(\Theta_{ij}|G) = \rho(\Theta_{ij}|G')$$

for all $j = 1, \ldots, q_i$.

Under assumption 1 to 4 Heckerman et al. [HGC95] showed the following result.

Theorem 2.3 (Heckerman, Geiger and Chickering [HGC95]) Under assumptions 1 through 4 we have that

$$P(B,T) = P(B) \times \prod_{i=1}^{n} \prod_{j=1}^{q_i} \left(\frac{\Gamma(N'_{ij})}{\Gamma(N_{ij} + N'_{ij})} \times \prod_{k=1}^{r_i} \frac{\Gamma(N_{ijk} + N'_{ijk})}{\Gamma(N'_{ijk})} \right)$$

where Γ is the Gamma function and P(B) represents the prior probability of the network B.

The theorem above induces the so-called *Bayesian Dirichlet* (BD) score defined as

$$\mathrm{BD}(B,T) = \log(P(B)) + \sum_{i=1}^{n} \sum_{j=1}^{q_i} \left(\log \left(\frac{\Gamma(N'_{ij})}{\Gamma(N_{ij} + N'_{ij})} \right) + \sum_{k=1}^{r_i} \log \left(\frac{\Gamma(N_{ijk} + N'_{ijk})}{\Gamma(N'_{ijk})} \right) \right).$$

Unfortunately, as Heckerman et al. recognized, specifying all N'_{ijk} for all i, j and k is formidable, to say the least. This makes the BD score unusable in practice. However, as we shall see next, there are some particular cases of the BD score that are useful.

K2 scoring function

One of the first Bayesian scoring functions, called K2, was proposed by Cooper and Herskovits [CH92] and it is a particular case of the BD score with the uninformative assignment $N'_{ijk} = 1$ (corresponding to zero pseudo-counts). Since $\Gamma(c) = (c-1)!$ when c is an integer, the K2 score can be expressed as follows:

$$\mathrm{K2}(B,T) = \log(P(B)) + \sum_{i=1}^{n}\sum_{j=1}^{q_i}\left(\log\left(\frac{(r_i-1)!}{(N_{ij}+r_i-1)!}\right) + \sum_{k=1}^{r_i}\log(N_{ijk}!)\right).$$

BDe scoring function

Heckerman et al [HGC95] turn around the problem of hyperparameter specification by considering two additional assumptions: likelihood equivalence and structure possibility. To introduce these hypotheses properly we require the concept of equivalent DAG's and some auxiliary notation.

Definition 2.4 (Equivalent directed acyclic graphs) Two directed acyclic graphs are *equivalent* if they can encode the same joint probability distributions.

Given a Bayesian network B, data T can be seen as a multinomial sample of the joint space $\mathbf{D}$ with parameters $\Theta_{\mathbf{D}} = \{\theta_{x_1\ldots x_n}\}_{x_i=1,\ldots,r_i,\, i\in 1\ldots n}$ where $\theta_{x_1\ldots x_n} = \prod_{i=1}^{n}\theta_{x_i|\Pi_{x_i}}$.

Assumption 5 (Likelihood equivalence) Given two directed acyclic graphs, G and G', such that $P(G) > 0$ and $P(G') > 0$, if G and G are equivalent then $\rho(\Theta_{\mathbf{D}}|G) = \rho(\Theta_{\mathbf{D}}|G')$.

Under the likelihood equivalence assumption it follows that for equivalent DAG's G and G' we have that $P(T|G) = P(T|G')$, that is, the data T does not help to discriminate equivalent DAG's.

Before introducing the structure possibility hypothesis we need the concept of complete DAG. The *skeleton* of any DAG is the undirected graph resulting from ignoring the directionality of every edge.

Definition 2.5 (Complete directed acyclic graph) A directed acyclic graph is said to be *complete* if its skeleton is complete.

In order to take advantage of the likelihood equivalence hypothesis the following assumption is needed.

Assumption 6 (Structure possibility) For any complete directed acyclic graph G, we have that $P(G) > 0$.

The likelihood equivalence hypothesis, when combined with the previous assumptions (Assumptions 1 to 4 and 6), introduces constraints in the Dirichlet exponents N'_{ijk}, as presented in the following result.

Theorem 2.6 (Heckerman, Geiger, Chickering [HGC95]) Suppose that $\rho(\Theta_{\mathbf{D}}|G)$ is Dirichlet with equivalent sample size N' for some complete directed acyclic graph G in $\mathbf{D}$. Then, for any Bayesian network B in $\mathbf{D}$, Assumptions 1 through 6 imply

$$P(B,T) = P(B) \times \prod_{i=1}^{n} \prod_{j=1}^{q_i} \left(\frac{\Gamma(N'_{ij})}{\Gamma(N_{ij} + N'_{ij})} \times \prod_{k=1}^{r_i} \frac{\Gamma(N_{ijk} + N'_{ijk})}{\Gamma(N'_{ijk})} \right)$$

where $N'_{ijk} = N' \times P(X_i = x_{ik}, \Pi_{X_i} = w_{ij}|G)$.

From Theorem 2.6 the hyperparameters N'_{ijk} can be easily computed from N' and $P(X_i = x_{ik}, \Pi_{X_i} = w_{ij}|G)$. To understand N' it is relevant to note that

$$N' = \sum_{x_1 \ldots x_n \in \mathbf{D}} N'_{x_1 \ldots x_n}, \tag{4}$$

where $N'_{x_1 \ldots x_n}$ are the hyperparameters of the Dirichlet $\rho(\Theta_{\mathbf{D}}|G)$ and so, N' corresponds to the sum of all pseudo-counts considered for $\rho(\Theta_{\mathbf{D}}|G)$.

The resulting scoring function is called *likelihood-equivalence Bayesian Dirichlet* (BDe) and its expression is identical to the BD expression. Similarly to the BD score, the BDe metric requires knowing $P(X_i = x_{ik}, \Pi_{X_i} = w_{ij}|G)$ for all i, j and k. This knowledge might not be elementary to find, which makes this score of little practical interest.

BDeu scoring function

A particular case of BDe, which is especially interesting, appears when

$$P(X_i = x_{ik}, \Pi_{X_i} = w_{ij}|G) = \frac{1}{r_i q_i},$$

that is, the prior network assigns a uniform probability to each configuration of $\{X_i\} \cup \Pi_{X_i}$ given the complete DAG G. The resulting score is called BDeu ("u" for uniform joint

distribution) and was originally proposed by Buntine [Bun91]. This score only depends on one parameter, the *equivalent sample size* N', and it is expressed as follows:

$$\text{BDeu}(B,T) = \log(P(B)) + \sum_{i=1}^{n}\sum_{j=1}^{q_i}\left(\log\left(\frac{\Gamma(\frac{N'}{q_i})}{\Gamma(N_{ij}+\frac{N'}{q_i})}\right) + \sum_{k=1}^{r_i}\log\left(\frac{\Gamma(N_{ijk}+\frac{N'}{r_iq_i})}{\Gamma(\frac{N'}{r_iq_i})}\right)\right).$$

The parameter N' expresses the strength of our prior belief in the uniformity of the conditional distributions of the network. Since there are no generally accepted rule to determine the hyperparameters $N'_{x_1\ldots x_n}$, there is no particular good candidate for N' given by Equation (4). In practice, the BDeu score is very sensitive with respect to the equivalent sample size N' and so, several values are attempted.

Regarding the term $\log(P(B))$ which appears in all the previous expressions (in BD, K2, BDe and BDeu scoring functions), it is quite common to assume a uniform distribution, except if, for some reason, we really prefer certain structures. When a uniform distribution is considered, the term $\log(P(B))$ becomes a constant and can be removed.

2.1.2 Information-theoretic scoring functions

Information-theoretic metrics are based on compression. In this context, the score of a Bayesian network B is related to the compression that can be achieved over the data T with an optimal code induced by B. Shannon's source coding theorem (or noiseless coding theorem) establishes the limits to possible data compression, and the operational meaning of the Shannon entropy.

Theorem 2.7 (Shannon source coding theorem) As the number of instances of an i.i.d. data tends to infinity, no compression of the data is possible into a shorter message length than the total Shannon entropy, without losing information.

There are several optimal codes that asymptotically achieve Shannon's limit, such as the Fano-Shannon code and the Huffman code. To construct these codes one requires as input a probability distribution over the data, which can be derived from a Bayesian network. So, given data T, one can score a Bayesian network B by the size of an optimal code, induced by the distribution B, when encoding T. This value is the *information content of T by B* and is

given by

$$\begin{aligned} L(T|B) &= -\log(P_B(T)) \\ &= -\sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} N_{ijk}\log(\theta_{ijk}) \quad (5) \\ &= -\sum_{i=1}^{n}\sum_{j=1}^{q_i} N_{ij}\sum_{k=1}^{r_i}\frac{N_{ijk}}{N_{ij}}\log(\theta_{ijk}). \quad (6) \end{aligned}$$

Gibb's inequality justifies the choice of parameters θ_{ijk} that minimizes $L(T|B)$.

Lemma 2.8 (Gibb's inequality) Let $P(x)$ and $Q(x)$ be two probability distributions over the same domain, then

$$\sum_x P(x)\log(Q(x)) \leq \sum_x P(x)\log(P(x)).$$

From the previous inequality, Equation (6) is minimized when

$$\theta_{ijk} = \frac{N_{ijk}}{N_{ij}}. \quad (7)$$

Thus, when fixing the DAG structure of a Bayesian network B, Equation (6) is minimized when $\theta_{ijk} = \frac{N_{ijk}}{N_{ij}}$. Clearly, $L(T|B)$ is minimal when the likelihood $P_B(T)$ of T given B is maximal, which means that the Bayesian network that induces a code that compresses T the most is precisely the Bayesian network that maximizes the probability of observing T.

LL scoring function

By applying a logarithm to the likelihood of T given B, we obtain $\log(P_B(T)) = -L(T|B)$ that is commonly called the *log-likelihood of T given B*. Observe that maximizing the log-likelihood is equivalent to minimizing the information content of T by B. This leads to defining the log-likelihood (LL) score [Bou95] in the following way:

$$\mathrm{LL}(B|T) = \sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} N_{ijk}\log\left(\frac{N_{ijk}}{N_{ij}}\right). \quad (8)$$

The LL score tends to favor complete network structures and it does not provide an useful representation of the independence assumptions of the learned network. This phenomenon of *overfitting* is usually avoided in two different ways. First, by limiting the number of parents per network variable. Second, by using some penalization factor over the LL score. We are particularly interested in the second approach which we discuss in the following.

MDL scoring function

The *minimum description length* (MDL) scoring function [LB94, Suz93] is an *Occam's razor* approach to fitting, preferring simple Bayesian networks over complex ones, and it is rigorously defined as:

$$\mathrm{MDL}(B|T) = \mathrm{LL}(B|T) - \frac{1}{2}\log(N)|B|, \tag{9}$$

where $|B|$ denotes the network complexity [Ris86], that is, the number of parameters in Θ for the network B, and it is given by:

$$|B| = \sum_{i=1}^{n}(r_i - 1)q_i.$$

In Equation (9), the first term measures how many bits are needed to describe data T based on the probability distribution P_B, whereas the second term represents the length of describing the network B, that is, it counts the number of bits needed to encode B, where $\frac{1}{2}\log(N)$ bits are used for each parameter in Θ.

AIC and BIC scoring functions

The measure of the quality of a Bayesian network can be computed in several different ways. This leads to a generalization of the MDL scoring function given by: $\phi(B|T) = \mathrm{LL}(B|T) - f(N)|B|$, where $f(N)$ is a non-negative penalization function. If $f(N) = 1$, we have the *Akaike Information Criterion* (AIC) scoring function [Aka74], that is,

$$\mathrm{AIC}(B|T) = \mathrm{LL}(B|T) - |B|. \tag{10}$$

If $f(N) = \frac{1}{2}\log(N)$, we have the *Bayesian Information Criterion* (BIC) score based on Schwarz Information Criterion [Sch78], which coincides with the MDL score. If $f(N) = 0$, we have the LL score.

NML scoring function

Recently, a new scoring function based on the MDL principle was proposed [KM07, RSKM08]. This new metric deserves a detailed presentation and justification for two reasons. First, it is very recent and poorly understood, although it is based on a well established idea [RT04]. Second, it has been presented in a completely different language and the presentation is spread in several papers which hinters its clear understanding.

The main idea behind the MDL principle is that to explain data T one should always choose the hypothesis with smallest description that generates T. This approach can be seen as the *Occam's razor* – the simplest explanation is the best – where simpler corresponds to smaller description. This leads to the problem of formalizing what is a *description* and its *length*. At first sight the answer seems to be the Kolmogorov complexity of T, that is, the size of the smallest program that generates T written in a fixed universal programming language. However, Kolmogorov complexity is undecidable, and so this approach is not feasible. Moreover, the size of the description depends on the chosen programming language. The problem seems hopeless, but when the hypotheses are probability distributions then it is possible to rely on information theory results.

Given data T and a set of probability distributions $\mathcal{H}$ that may be used to describe T, we take the *length of describing T with H* to be the sum $L(T|H) + L(H)$, where $L(T|H)$ is the length (in bits) of the description of T when encoded with H and $L(H)$ is the length of the description of H. There is absolutely no doubt that $L(T|H)$ should be the size of the (asymptotic) Shannon-Fano code (or any optimal code, such as the Huffman code) for encoding T given H, or (minus the) log-likelihood of T given H, that is, $L(T|H) = -\mathrm{LL}(H|T) = -\log P(T|H)$ where $P(T|H)$ is the probability of sampling T with distribution H. However, defining $L(H)$ has never been consensual. Indeed, by looking at the Bayesian network scoring functions like BIC/MDL and AIC (see Equations (9) and (10)), one gets an evidence of how is hard to come up with $L(H)$. Both these scores agree in setting $L(T|H) = -\mathrm{LL}(H|T)$ but AIC sets $L(H) = |B|$ whereas BIC/MDL sets $L(H) = \frac{1}{2}\log(N)|B|$. Nevertheless, it is easy to find arguments against both choices for $L(H)$.

Actually, using $|B|$ in the expression of the complexity of a Bayesian network is, in general, an error. The reason is that the parameters of a Bayesian network are conditional distributions. Thus, if there are probabilities in Θ taking value 0, they do not need to appear in the description of Θ. Moreover, the same distribution (or probability value) might occur several times in Θ leading to patterns that can be exploited to compress Θ significantly. As a pathological example consider a Bayesian network with two variables X and Y such that Y depends on X. Moreover, assume that X is a Bernoulli and Y ranges over 1000 values. It is easy to see that $|\Theta| = 1 + 2 \times 999$. However, assume that $P(Y = 0|X = 1) = 1$, $P(Y = n|X = 0) = 1/1000$ for $0 \leq n \leq 999$ and $P(X = 0) = \frac{1}{2}$. It is clear that we can

describe these distributions very concisely, however, if the distributions for X and Y were without patterns we might need to describe $1 + 2 \times 999$ different real numbers.

There have been attempts to correct $L(H)$ in the AIC and BIC/MDL scores along these lines [ZK04], however, these works are supported more on empirical evidence than on theoretical results. The main breakthrough in the community in the direction of computing $L(H)$ was to consider normalized minimum likelihood codes [RT04].

The idea behind normalized minimum likelihood codes is the same of universal coding. Suppose an encoder is about to observe data T which he plans to compress as much as possible. The encoder has a set of candidate codes $\mathcal{H}$ and he believes one of these codes will allow to compress the incoming data significantly, however, he has to choose the code before observing the data. It is clear that in general there is no code which, no mater what incoming data T is, will always mimic the best code for T. So what is the best thing that the encoder can do? There are simple solutions to this problem when $\mathcal{H}$ is finite, however, this is not the case for Bayesian networks.

To answer this question we recast the problem in a stochastic wording. Given a set of probability distributions $\mathcal{H}$ the encoder thinks that there is one distribution $H \in \mathcal{H}$ that will assign high likelihood (low code length) to the incoming data T of fixed size N. Therefore, we will like to design a code that for all T it will compress T as close as possible to the code associated to $H \in \mathcal{H}$ that maximizes the likelihood of T. We call to this $H \in \mathcal{H}$ the *best-fitting hypothesis*. We can compare the *performance of a distribution H w.r.t. H' of modeling T of size N* by computing

$$-\log(P(T|H)) + \log(P(T|H')).$$

This value computes the additional number of bits needed to encode T with H, as compared to the number of bits that had been needed if we had used H'. Given a set of probability distributions $\mathcal{H}$ and a distribution $\overline{H}$ not necessarily in $\mathcal{H}$, the *regret of $\overline{H}$ relative to $\mathcal{H}$ for T of size N* is

$$-\log(P(T|\overline{H})) - \min_{H \in \mathcal{H}}(-\log(P(T|H))),$$

which corresponds to computing the performance of $\overline{H}$ compared to the best fitting hypothesis (in terms of log likelihood) in $\mathcal{H}$. In many practical cases, given a set of hypothesis $\mathcal{H}$ and data T, we are always able to find the $H_{\mathcal{H}}(T) \in \mathcal{H}$ that minimizes $-\log(P(T|H))$. For instance, when $\mathcal{H}$ is the set of tree-structured Bayesian networks there is an algorithm to compute an

optimal Bayesian network $H_{\mathcal{H}}(T)$ relative to the LL score (c.f. Section 2.3). In this case it is possible to rewrite the regret of $\overline{H}$ relative to $\mathcal{H}$ for T of size N as

$$-\log(P(T|\overline{H})) + \log(P(T|H_{\mathcal{H}}(T))).$$

The notion of regret is relative to some data T, however, it is possible to compare the performance of the hypothesis taking into account all data of fixed size N. The idea is to take the worst-case approach over all data of size N. Formally, the *worst-case regret of* $\overline{H}$ *relative to* $\mathcal{H}$ *for data of size* N is given by

$$\max_{T:|T|=N}(-\log(P(T|\overline{H})) + \log(P(T|H_{\mathcal{H}}(T)))).$$

Finally, the universal distribution for $\mathcal{H}$ is one that minimizes the worst-case regret.

Definition 2.9 (Universal distribution) Let $\mathcal{H}$ be a set of probability distributions for which it is always possible to find the distribution $H_{\mathcal{H}}(T) \in \mathcal{H}$ that minimizes $-\log(P(T|H))$. The *universal distribution relative to* $\mathcal{H}$ *for data of size* N is the probability distribution $H_{\mathcal{H}}(N)$ such that

$$H_{\mathcal{H}}(N) = \min_{\overline{H}} \max_{T:|T|=N}(-\log(P(T|\overline{H})) + \log(P(T|H_{\mathcal{H}}(T)))),$$

where the minimum is taken over all distributions on the data space of size N.

It is possible to compute the universal distribution relative to $\mathcal{H}$ when $\mathcal{H}$ has finite parametric complexity. Before presenting this result, we define the *parametric complexity of* $\mathcal{H}$ *for data of size* N to be

$$\mathbf{C}_N(\mathcal{H}) = \log\left(\sum_{T:|T|=N} P(T|H_{\mathcal{H}}(T))\right).$$

Theorem 2.10 (Shtakov [Sht87]) Let $\mathcal{H}$ be a set of probability distributions such that $\mathbf{C}_N(\mathcal{H})$ is finite. Then, the universal distribution relative to $\mathcal{H}$ for data of size N is given by

$$P_{\mathcal{H}}^{\mathrm{NML}}(T) = \frac{P(T|H_{\mathcal{H}}(T))}{\sum_{T':|T'|=N} P(T'|H_{\mathcal{H}}(T'))}.$$

The distribution $P_{\mathcal{H}}^{\mathrm{NML}}(T)$ is called the *normalized maximum likelihood* (NML) distribution. We now show how the NML distribution can be used to set up a scoring function for Bayesian networks.

Suppose that we have two potential models for data T, that is, two sets of probability distributions $\mathcal{H}_0$ and $\mathcal{H}_1$. In the context of Bayesian networks, consider for example that we want to know which is the best set of parents Π_X for a node X: $\{Y\}$ or $\{Y, Z\}$. The MDL principle states we should pick $\mathcal{H}_j$ that maximizes the normalized maximum likelihood $P^{\text{NML}}_{\mathcal{H}_j}(T)$, that is, we should pick $\mathcal{H}_j$ that maximizes

$$\begin{aligned} \log(P^{\text{NML}}_{\mathcal{H}_j}(T)) &= \log(P(T|H_{\mathcal{H}_j(T)})) - \mathbf{C}_N(\mathcal{H}_j) \\ &= \text{LL}(H_{\mathcal{H}_j(T)}|T) - \mathbf{C}_N(\mathcal{H}_j). \end{aligned} \tag{11}$$

The quantity $-\log(P^{\text{NML}}_{\mathcal{H}_j}(T))$ is called the *stochastic complexity of data T relative to $\mathcal{H}_j$*. Observe that maximizing the normalized maximum likelihood is equivalent to minimizing the stochastic complexity. By comparing Equation (11) with the BIC/MDL score (9) and AIC score (10), it is clear that we are replacing the terms measuring the complexity of the network by the parametric complexity.

We are now able to establish a Bayesian network scoring function based on the normalized maximum likelihood. Let $\mathcal{B}_G$ denote the set of all Bayesian networks with network structure G. For a fixed a network structure G, the normalized maximum likelihood (NML) score is defined as

$$\text{NML}(B|T) = \text{LL}(B|T) - \mathbf{C}_N(\mathcal{B}_G). \tag{12}$$

From the definition of parametric complexity, we have that

$$\begin{aligned} \mathbf{C}_N(\mathcal{B}_G) &= \log\left(\sum_{T:|T|=N} P(T|B_T)\right) \\ &= \log\left(\sum_{T:|T|=N} \prod_{t=1}^{N} P_{B_T}(\mathbf{y}_t)\right) \\ &= \log\left(\sum_{T:|T|=N} \prod_{t=1}^{N} \prod_{i=1}^{n} P_{B_T}(\mathbf{y}_{ti}|\mathbf{y}_{t\Pi_{X_i}})\right), \end{aligned} \tag{13}$$

where B_T is the Bayesian network with maximal log-likelihood for data T. Unfortunately, there is no hope for computing $\mathbf{C}_N(\mathcal{B}_G)$ efficiently, since it involves an exponential sum over all possible data of size N. Moreover, the score presented in this way would not be decomposable, which, as discussed in the next section, would not allow the use of efficient local search methods. The idea in [RSKM08] is to approximate Equation (13) by considering

only the contribution to the parametric complexity of the multinomial distributions associated to each variable given a parent configuration, that is,

$$\begin{aligned}
\mathbf{fC}_T(\mathcal{B}_G) &= \log\left(\prod_{i=1}^{n}\prod_{j=1}^{q_i}\sum_{T\in r_i^{N_{ij}}}\prod_{t=1}^{N_{ij}}\hat{P}_T(\mathbf{y}_t)\right) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{q_i}\log\left(\sum_{T\in r_i^{N_{ij}}}\prod_{t=1}^{N_{ij}}\hat{P}_T(\mathbf{y}_t)\right) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{q_i}\mathbf{C}_{N_{ij}}(\mathcal{M}_{r_i}),
\end{aligned}$$

where $r_i^{N_{ij}}$ is the set of all sequences of size N_{ij} written with an alphabet with r_i symbols, $\hat{P}_T(\mathbf{y}_t)$ is the frequency of $\mathbf{y}_t$ in T and $\mathcal{M}_{r_i}$ is the set of all multinomial distributions with r_i parameters. It is easy to see that the score given by

$$\mathrm{fNML}(B|T) = \mathrm{LL}(B|T) - \mathbf{fC}_T(\mathcal{B}_G)$$

decomposes over the network structure, and that maximizing it is equivalent to maximizing

$$\mathrm{fNML}(B|T) = \sum_{i=1}^{n}\sum_{j=1}^{q_i}\left(\sum_{k=1}^{r_i}N_{ijk}\log\left(\frac{N_{ijk}}{N_{ij}}\right) - \mathbf{C}_{N_{ij}}(\mathcal{M}_{r_i})\right). \tag{14}$$

Finally, we note that from the definition, the computation of $\mathbf{C}_{N_{ij}}(\mathcal{M}_{r_i})$ seems exponential in N_{ij}, since it involves an exponential sum over all possible data of size N_{ij}. However, it was recently proposed [KM07] a linear-time algorithm for computing the stochastic complexity in the case of N_{ij} observations of a single multinomial random variable. For that purpose an elegant recursion formula was proposed based on the mathematical technique of generating functions. The algorithm for computing the multinomial parametric complexity is presented in Algorithm 1.

Algorithm 1 Multinomial parametric complexity

Compute $\mathbf{C}_{N_{ij}}(\mathcal{M}_{r_i}) = \log(\mathcal{C}(r_i, N_{ij}))$ where $\mathcal{C}(\ell, m)$ is obtained by the following recurrence:

1. $\mathcal{C}(1, m) = 1$.
2. $\mathcal{C}(2, m) = \sum_{h_1+h_2=m}\frac{m!}{h_1!h_2!}\left(\frac{h_1}{m}\right)^{h_1}\left(\frac{h_2}{m}\right)^{h_2}$.
3. If $(\ell > 2)$ then $\mathcal{C}(\ell, m) = \mathcal{C}(\ell-1, m) + \frac{m}{\ell-2}\,\mathcal{C}(\ell-2, m)$.

Since the NML score (Equation (12)) is intractable, we will only consider the fNML score (Equation (14)), and for the sake of simplicity we will drop the 'f' and write just NML for fNML.

MIT scoring function

A scoring function based on mutual information, called *mutual information tests* (MIT) score, was proposed by de Campos [dC06] and its expression is given by:

$$\mathrm{MIT}(B|T) = \sum_{\substack{i=1 \\ \Pi_{X_i} \neq \emptyset}}^{n} \left(2NI(X_i; \Pi_{X_i}) - \sum_{j=1}^{s_i} \chi_{\alpha, l_{i\sigma_i^*(j)}} \right)$$

where $I(X_i; \Pi_{X_i})$ is the mutual information between X_i and Π_{X_i} in the network which measures the degree of interaction between each variable and its parents. This measure is, however, penalized by a term related to the Pearson χ^2 test of independence. This term attempts to re-scale the mutual information values in order to prevent them from systematically increasing as the number of variables in Π_{X_i} does, even when newly added variables in Π_{X_i} are independent of X_i. In this penalization component, α is a free parameter representing the confidence level associated with the statistical test (for instance, 0.90, 0.95 or 0.99) and, $\sigma_i^* = (\sigma_i^*(1), \ldots, \sigma_i^*(s_i))$ denotes any permutation of the index set $(1, \ldots, s_i)$ of the variables in $\Pi_{X_i} = \{X_{i1}, \ldots, X_{is_i}\}$ satisfying $r_{i\sigma_i^*(1)} \geq r_{i\sigma_i^*(2)} \geq \cdots \geq r_{i\sigma_i^*(s_i)}$, where r_{ij} represents the number of possible configurations when the parent set of X_i is restricted only to X_j. Moreover, the number of degrees of freedom $l_{i\sigma_i^*(j)}$ is given by:

$$l_{i\sigma_i^*(j)} = \begin{cases} (r_i - 1)(r_{i\sigma_i^*(j)} - 1) \prod_{k=1}^{j-1} r_{i\sigma_i^*(k)} & j = 2, \ldots, s_i \\ (r_i - 1)(r_{i\sigma_i^*(j)} - 1) & j = 1. \end{cases}$$

2.2 Decomposability and score equivalence

For efficiency purposes, a scoring function being used in the context of a *score+search* method needs to have the property of decomposability.

Definition 2.11 (Decomposable scoring function) A scoring function ϕ is *decomposable* if the score assigned to each network decompose over the network structure in such a way that it can be expressed as a sum of local scores that depends only on each node and its

parents, that is, scores of the following form:

$$\phi(B,T) = \sum_{i=1}^{n} \phi_i(\Pi_{X_i}, T).$$

When considering the search in the space of equivalence classes of network structures score equivalent scoring functions are particularly interesting. The definition of score-equivalent scoring functions need some background in Bayesian network theory which is introduced next.

Two variables X and Y are *adjacent* if there is an edge between X and Y.

Definition 2.12 (v-structure) In a directed acyclic graph, a *v-structure* is a local dependency $X \rightarrow Z \leftarrow Y$ such that X and Y are not adjacent.

Theorem 2.13 (Verma and Pearl [VP90]) Two directed acyclic graphs are equivalent if and only if they have the same skeleton and the same v-structures.

By Theorem 2.13, all trees network structures with the same skeleton are equivalent, regardless from the direction of the edges.

Because DAG equivalence is reflexive, symmetric, and transitive, it defines a set of equivalence classes over DAG's. One way to represent the equivalence class of equivalent DAG's is by the means of a partially directed acyclic graph.

Definition 2.14 (Partially directed acyclic graph) A *partially directed acyclic graph* is a graph which contains both directed and undirected edges, with no directed cycle in its directed subgraph.

From Theorem 2.13, it follows that a PDAG containing a directed edge for every edge participating in a v-structure, and an undirected edge for every other edge, uniquely identifies an equivalence class of DAG's. There may be many other PDAG's, however, that correspond to the same equivalence class. For example, any DAG interpreted as a PDAG can be used to represent its own equivalence class.

Definition 2.15 (Compelled edge) A directed edge $X \rightarrow Y$ is *compelled* in a directed acyclic graph G if for every directed acyclic graph G' equivalent to G, $X \rightarrow Y$ exists in G'.

By Theorem 2.13, all edges participating in a v-structure are compelled. Not every compelled edge, however, necessarily participates in a v-structure. For example, the edge $Z \rightarrow W$ in the DAG with edges $E = \{X \rightarrow Z, Z \rightarrow W, Y \rightarrow Z, Y \rightarrow U\}$ is compelled. Moreover, for any edge e in G, if e is not compelled in G, then e is *reversible* in G. In that case, there exists some DAG G' equivalent to G in which e has opposite direction.

Definition 2.16 (Essential graph) An *essential graph*, denoting an equivalence class of directed acyclic graphs, is the partially directed acyclic graph consisting of a directed edge for every compelled edge in the equivalence class, and an undirected edge for every reversible edge in the equivalence class.

Essential graphs are used to represent equivalent class of network structures during Bayesian network learning. The essential graph of a tree network structure is its skeleton.

Definition 2.17 (Score equivalent scoring function) A scoring function ϕ is *score equivalent* if it assigns the same score to all directed acyclic graphs that are represented by the same essential graph.

All interesting scoring functions in the literature are decomposable, since it is unfeasible to learn undecomposable scores. LL, AIC, BIC/MDL are decomposable and score equivalent, whereas K2, BD, BDe, BDeu, NML and MIT are decomposable but not score equivalent. The score equivalence property is mandatory when searching in the space of equivalence classes of network structures. However, in general, it does not seem to be an important property. Indeed, non-score-equivalent scoring functions typically perform better than score equivalent ones [dC06, YC02].

2.3 Chow-Liu tree learning algorithm

A *tree Bayesian network* is a Bayesian network where the underlying DAG is a directed tree. Finding the tree Bayesian network that maximizes the LL score given data T can be done in polynomial time by the Chow-Liu tree learning algorithm [CL68].

In order to understand how to solve the learning problem for tree Bayesian networks we need to reformulate the $\mathrm{LL}(B|T)$, presented in Equation (8), using mutual information

[Bou95]. Applying Equation (1) to the log-likelihood given by

$$\mathrm{LL}(B|T) = \sum_{t=1}^{N} \log(P_B(\mathbf{y}_t)) \tag{15}$$

we obtain that

$$\mathrm{LL}(B|T) = N \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} \hat{P}_T(X_i = x_{ik}, \Pi_{X_i} = w_{ij}) \log(\theta_{ijk}) \tag{16}$$

where $\hat{P}_T$ is the *empirical distribution* defined by the frequency of each possible configuration $\{X_i\} \cup \Pi_{X_i}$ in T. More precisely,

$$\hat{P}_T(X_i = x_{ik}, \Pi_{X_i} = w_{ij}) = \frac{N_{ijk}}{N}.$$

From Gibb's inequality (Lemma 2.8), Equation (16) is maximized when

$$\theta_{ijk} = \hat{P}_T(X_i = x_{ik} | \Pi_{X_i} = w_{ij}) = \frac{N_{ijk}}{N_{ij}}. \tag{17}$$

Thus, by fixing the underlying DAG of a Bayesian network B, the choice of parameters θ_{ijk} that maximizes $\mathrm{LL}(B|T)$ is given by Equation (17). This means that it is enough to search for the underlying DAG of B that maximizes the LL score. From this point on it is assumed that the parameters of B fulfill Equation (17) and, so, Equation (16) can be rewritten as follows:

$$\begin{aligned} \mathrm{LL}(B|T) &= -N \sum_{i=1}^{n} H_{\hat{P}_T}(X_i | \Pi_{X_i}) \\ &= N \sum_{i=1}^{n} I_{\hat{P}_T}(X_i; \Pi_{X_i}) - N \sum_{i=1}^{n} H_{\hat{P}_T}(X_i), \end{aligned} \tag{18}$$

where $I_{\hat{P}_T}(X_i; \Pi_{X_i})$ is the mutual information between X_i and Π_{X_i}, and $H_{\hat{P}_T}$ is the entropy both given by the empirical distribution. Observe that the right-hand side of (18) has two terms and only the first depends on the Bayesian network B, hence, maximizing $\mathrm{LL}(B|T)$ resumes to maximize

$$\sum_{i=1}^{n} I_{\hat{P}_T}(X_i; \Pi_{X_i}) = \sum_{\substack{i=1 \\ i \neq R}}^{n} I_{\hat{P}_T}(X_i; X_{\pi(i)})$$

where R is the root of the tree Bayesian network B and $\pi(i)$ is the index of the parent variable of X_i, that is, $\Pi_{X_i} = \{X_{\pi(i)}\}$ for $i \neq R$. Recall that the *mutual information of two random vectors* is given by

$$I(\mathbf{X}; \mathbf{Y}) = \sum_{\mathbf{x},\mathbf{y}} P(\mathbf{x}, \mathbf{y}) \log \frac{P(\mathbf{x}, \mathbf{y})}{P(\mathbf{x})P(\mathbf{y})}. \tag{19}$$

The main idea of the algorithm to learn tree Bayesian networks is to consider a complete weighted undirected graph, where each undirected edge between X_i and X_j is weighted with the mutual information between X_i and X_j. Given this, the problem reduces to determining a maximal weighted (undirected) spanning tree. After computing such spanning tree, a direction has to be assigned to each edge of the tree. This is done by choosing an arbitrary node as the tree root and then setting the direction of all edges to be outward from it. The detail of the algorithm is depicted in Algorithm 2.

Algorithm 2 Chow-Liu tree learning algorithm, for the LL score

1. Compute the mutual information $I_{\hat{P}_T}(X_i; X_j)$ between each pair of attributes, with $i \neq j$ and $i, j \leq n$, given by Equation (19).
2. Build a complete undirected graph with attributes $X_1, \ldots, X_n$ as nodes. Annotate the weight of the edge connecting X_i to X_j by $I_{\hat{P}_T}(X_i; X_j)$.
3. Build a maximal weighted (undirected) spanning tree.
4. Transform the resulting undirected tree to a directed one by choosing a root variable and setting the direction of all edges to be outward from it and return the resulting tree.

The resulting directed tree is called *Chow-Liu tree* or *optimal branching*. Chow-Liu [CL68] showed that Algorithm 2 is linear on the size of the data T and quadratic on the number of variables of the Bayesian network.

Theorem 2.18 (Chow and Liu [CL68]) Let T be a collection of N instances of $X_1, \ldots, X_n$. Algorithm 2 constructs an optimal branching B that maximizes $\mathrm{LL}(B|T)$ in $O(n^2 N)$ time.

2.4 Extending Chow-Liu tree learning algorithm

The Chow-Liu tree learning algorithm was originally proposed for maximizing the LL score but it can be easily adapted to deal with any scoring function that is decomposable and/or score equivalent.

According to Heckerman et al [HGC95], finding an optimal branching for decomposable and score equivalent scoring functions reduces to weighting each undirected edge between X_i and X_j by $\phi_j(\{X_i\}, T) - \phi_j(\emptyset, T)$, which is equal to $\phi_i(\{X_j\}, T) - \phi_i(\emptyset, T)$ by score equivalence of ϕ, and to find a maximal weighted (undirected) spanning tree. The detailed algorithm for learning tree Bayesian networks for decomposable ans score-equivalent scoring functions is presented in Algorithm 3.

Algorithm 3 Learning tree Bayesian networks, for any decomposable and score equivalent ϕ–score

1. Compute $\phi_j(\{X_i\}, T) - \phi_j(\emptyset, T)$ between each pair of attributes X_i and X_j, with $i \neq j$ and $i, j \leq n$.
2. Build a complete undirected graph with attributes $X_1, \ldots, X_n$ as nodes. Annotate the weight of the edge connecting X_i to X_j by the value computed in the previous step.
3. Build a maximal weighted (undirected) spanning tree.
4. Transform the resulting undirected tree to a directed one by choosing a root variable and setting the direction of all edges to be outward from it and return the resulting tree.

Learning an optimal branching for scoring functions that are only decomposable, but not score equivalent, can also be done in polynomial time [HGC95]. In this case, however, an edge between X_i and X_j may score differently depending on its direction, and so a directed spanning tree must be found (instead of an undirected one). The idea is to weight each directed edge from X_i to X_j with $\phi_j(\{X_i\}, T) - \phi_j(\emptyset, T)$ and then find an optimal directed spanning tree with Edmonds' algorithm [Edm67, Law76]. The detailed algorithm for learning tree Bayesian networks for scoring functions that are only decomposable, but not score-equivalent, is presented in Algorithm 4.

Algorithm 4 Learning tree Bayesian networks, for any decomposable and non-score equivalent ϕ–score

1. Compute $\phi_j(\{X_i\}, T) - \phi_j(\emptyset, T)$ for each edge from X_i to X_j, with $i \neq j$ and $i, j \leq n$.
2. Build a complete directed graph with attributes $X_1, \ldots, X_n$ as nodes. Annotate the weight of the edge from X_i to X_j by the value computed in the previous step.
3. Build a maximal weighted directed spanning tree and return it.

3 Bayesian network classifiers

Bayesian networks have been widely used in the context of classification [SZ06, GD04, FGG97, DH73]. Herein, we introduce the concept of Bayesian network classifier and then present the Tree Augmented Naive Bayes [FGG97] classifier.

Definition 3.1 (Bayesian network classifier) A *Bayesian network classifier* is a Bayesian network where $\mathbf{X} = (X_1, \ldots, X_n, C)$. The variables $X_1, \ldots, X_n$ are called *attributes* and C is called the *class variable*. Moreover, the graph structure G is such that the class variable has no parents, that is, $\Pi_C = \emptyset$, and all attributes have at least the class variable as parent, that

is, $C \in \Pi_{X_i}$. The corresponding classifier is defined as

$$\arg\max_C P_B(C|X_1, \ldots, X_n).$$

We therefore reformulate the model to make it more tractable. Using the definition of conditional probability and Equation (1) leads to the following classifier:

$$\arg\max_C P_B(C) \prod_{i=1}^{n} \theta_{X_i|\Pi_{X_i}}.$$

Informally, the problem of learning a Bayesian network classifier can be recasted as the problem of learning a Bayesian network where all attributes have the class variable as parent.

3.1 Tree augmented naive Bayesian network classifier

The tree augmented naive (TAN) Bayesian network was first proposed by Friedman et al [FGG97]. It is a classifier which restricts correlations between the attributes of the network to a tree structure.

Definition 3.2 (Tree augmented naive Bayesian network) A *tree augmented naive Bayesian network* (TAN) [FGG97] is a Bayesian network classifier where there exists a root $R \in \{1, \ldots, n\}$ such that $\Pi_{X_R} = \{C\}$ and $\Pi_{X_i} = \{C, X_j\}$ for all $1 \leq i \leq n$ with $i \neq R$.

In order to understand how to solve the learning problem for TAN Bayesian networks we need to reformulate the $\mathrm{LL}(B|T)$ using mutual information as in (18). With TAN models however we have to consider the class variable and so,

$$\begin{aligned} \mathrm{LL}(B|T) &= -N \sum_{i=1}^{n} H_{\hat{P}_T}(X_i|\Pi_{X_i}, C) \\ &= N \sum_{i=1}^{n} I_{\hat{P}_T}(X_i; \Pi_{X_i}, C) - N(H_{\hat{P}_T}(C) + \sum_{i=1}^{n} H_{\hat{P}_T}(X_i)) \end{aligned} \tag{20}$$

Observe that the right-hand side of (20) has two terms and only the first depends on the Bayesian network B, hence, maximizing $\mathrm{LL}(B|T)$ resumes to maximize

$$\sum_{i=1}^{n} I_{\hat{P}_T}(X_i; \Pi_{X_i}, C). \tag{21}$$

We can simplify (21) for TAN models using the chain law for mutual information,

$$I(\mathbf{X}; \mathbf{Y}, \mathbf{Z}) = I(\mathbf{X}; \mathbf{Z}) + I(\mathbf{X}; \mathbf{Y}|\mathbf{Z}),$$

and derive

$$\sum_{i=1}^{n} I_{\hat{P}_T}(X_i; C) + \sum_{\substack{i=1 \\ i \neq R}}^{n} I_{\hat{P}_T}(X_i; X_{\pi(i)}|C). \tag{22}$$

Finally, note that the first term of (22) does not depend on the choice of the parents $\pi(i)$, therefore, maximizing $\mathrm{LL}(B|T)$ is equivalent to maximize

$$\sum_{\substack{i=1 \\ i \neq R}}^{n} I_{\hat{P}_T}(X_i; X_{\pi(i)}|C). \tag{23}$$

Recall that the *conditional mutual information* is given by

$$I(\mathbf{X}; \mathbf{Y}|\mathbf{Z}) = \sum_{\mathbf{x},\mathbf{y},\mathbf{z}} P(\mathbf{x}, \mathbf{y}, \mathbf{z}) \log \frac{P(\mathbf{x}, \mathbf{y}|\mathbf{z})}{P(\mathbf{x}|\mathbf{z})P(\mathbf{y}|\mathbf{z})}. \tag{24}$$

It is now easy to find the TAN that maximizes the LL score for some data T. The main idea is to consider a complete weighted undirected graph, where each edge between X_i and X_j is weighted with the conditional mutual information between X_i and X_j given the class variable C. Given this, the problem reduces to determining a maximal weighted (undirected) spanning tree. After computing such spanning tree, a direction has to be assigned to each edge of the tree. This is done by choosing an arbitrary attribute as the tree root and then setting the direction of all edges to be outward from it. The TAN Bayesian network classifier is then built by adding a node labeled by C, and adding an arc from C to each tree node. The detail of the algorithm is depicted in Algorithm 5.

Algorithm 5 Learning TAN Bayesian network classifiers, for the LL score

1. Compute $I_{\hat{P}_T}(X_i; X_j|C)$ between each pair of attributes, with $i \neq j$ and $i, j \leq n$, given by Equation (24).
2. Build a complete undirected graph with attributes $X_1, \ldots, X_n$ as nodes. Annotate the weight of the edge connecting X_i to X_j by $I_{\hat{P}_T}(X_i; X_j|C)$.
3. Build a maximal weighted (undirected) spanning tree.
4. Transform the resulting undirected tree to a directed one by choosing a root variable and setting the direction of all edges to be outward from it.
5. Construct a TAN Bayesian network classifier by adding a node labeled by C and adding an arc from C to each X_i, $i \leq n$.

The proof of soundness of Algorithm 5 follows from the derivation that led to Equation (23) and from the fact that we are computing a maximal weighted spanning tree. Since the step that consumes asymptotically more time is weighting the edges, Algorithm 5 is linear on the size of the data T and quadratic on the number of variables of the Bayesian network.

Theorem 3.3 (Friedman, Geiger and Goldszmidt [FGG97]) Let T be a collection of N instances of $X_1, \ldots, X_n$. Algorithm 5 constructs a TAN Bayesian network B that maximizes $\mathrm{LL}(B|T)$ in $\mathrm{O}(n^2 N)$ time.

3.2 Extending tree augmented naive Bayesian network classifier

The TAN learning algorithm was originally proposed for maximizing the LL score but it can be easily adapted to deal with any scoring function that is decomposable and score equivalent.

According to Heckerman et al [HGC95], finding an optimal TAN classifier for decomposable and score equivalent scoring functions reduces to weighting each undirected edge between X_i and X_j by $\phi_j(\{X_i, C\}, T) - \phi_j(\{C\}, T)$, which is equal to $\phi_i(\{X_j, C\}, T) - \phi_i(\{C\}, T)$ by score equivalence of ϕ, and to find a maximal weighted (undirected) spanning tree. The detailed algorithm for learning TAN Bayesian network classifiers for decomposable ans score-equivalent scoring functions is presented in Algorithm 6.

Algorithm 6 Learning TAN Bayesian network classifiers, for any decomposable and score equivalent ϕ–score

1. Compute $\phi_j(\{X_i, C\}, T) - \phi_j(\{C\}, T)$ between each pair of attributes X_i and X_j, with $i \neq j$ and $i, j \leq n$.
2. Build a complete undirected graph with attributes $X_1, \ldots, X_n$ as nodes. Annotate the weight of the edge connecting X_i to X_j by the value computed in the previous step.
3. Build a maximal weighted (undirected) spanning tree.
4. Transform the resulting undirected tree to a directed one by choosing a root variable and setting the direction of all edges to be outward from it.
5. Construct a TAN Bayesian network classifier by adding a node labeled by C and adding an arc from C to each X_i, $i \leq n$.

Learning an optimal TAN classifier for scoring functions that are only decomposable, but not score equivalent, can also be done in polynomial time [HGC95]. In this case, however, an edge between X_i and X_j may score differently depending on its direction, and so a directed spanning tree must be found (instead of an undirected one). The idea is to weight each directed edge from X_i to X_j with $\phi_j(\{X_i, C\}, T) - \phi_j(\{C\}, T)$ and then find an optimal directed spanning tree with Edmonds' algorithm [Edm67, Law76]. The detailed algorithm for learning TAN Bayesian network classifiers for scoring functions that are only decomposable, but not score-equivalent, is presented in Algorithm 7.

Algorithm 7 Learning TAN Bayesian network classifiers, for any decomposable and non-score equivalent ϕ–score

1. Compute $\phi_j(\{X_i, C\}, T) - \phi_j(\{C\}, T)$ for each edge from X_i to X_j, with $i \neq j$ and $i, j \leq n$.
2. Build a complete directed graph with attributes $X_1, \ldots, X_n$ as nodes. Annotate the weight of the edge from X_i to X_j by the value computed in the previous step.
3. Build a maximal weighted directed spanning tree.
4. Construct a TAN Bayesian network classifier by adding a node labeled by C and adding an arc from C to each X_i, $i \leq n$.

4 Experiments on the UCI repository data

We implemented the Chow-Liu tree learning algorithm and its extensions in Mathematica 6.0, on top of the Combinatorica package [PS03]. The package was extended with a non-recursive, and efficient, version of Edmonds' algorithm to build a maximal directed spanning tree of a strongly connected weighted directed graphs.[2] A package to learn Bayesian network classifiers was implemented, and at the moment it allows to learn an optimal TAN classifier for any score discussed in this work. The package also contains the entropy based discretization algorithm [FI93] to deal with continuous datasets.

We ran our experiments on several datasets from the UCI repository [NHBM98]. The chosen datasets are presented in Table 1.

The information-theoretic scores used in the experiments were the LL, BIC/MDL, NML and the MIT with a 99% confidence level. The Bayesian scores considered were the K2 and BDeu with equivalent sample sizes 1, 4 and 16. The accuracy of each classifier is based on the percentage of successful predictions on the test sets of each dataset. Accuracy was measured via the *holdout method* for larger training sets, and via *5-fold cross-validation* for smaller ones [Koh95]. Results are presented in Table 2, where the accuracy is annotated by a 95% confidence interval.

In general all scores perform similarly. The Bayesian scores hardly distinguish among each other, except for the soybean-large dataset. For large datasets Bayesian scores perform well. The only significant result is the performance of NML in the soybean-large dataset, which is impressive. The NML score also performs well in vote and glass, which indicates that this is a good score for small datasets. These conclusions were taken from the results presented in

[2] Impressively, although package Combinatorica has an extensive library of graph algorithms, it misses this important and non-trivial algorithm.

Dataset	n	$\lvert D_C \rvert$	Train	Test
letter	16	26	15000	5000
satimage	36	6	4435	2000
chess	36	2	2130	1066
vehicle	18	4	846	CV-5
diabetes	8	2	768	CV-5
soybean-large	35	19	562	CV-5
vote	16	2	435	CV-5
heart	13	2	270	CV-5
glass	9	7	214	CV-5
iris	4	3	150	CV-5
lymphography	18	4	148	CV-5
hepatitis	19	2	80	CV-5

Table 1: Description of the datasets used in the experiments. The datasets are presented by decreasing size of the training set.

the following table and figures.

Data set	LL	BIC/MDL	NML	MIT(0.99)	K2	BDeu(1)	BDeu(4)	BDeu(16)
letter	78.48 ± 1.13	77.96 ± 1.15	75.02 ± 1.20	77.98 ± 1.15	$\mathbf{82.14 \pm 1.06}$	$\mathbf{82.25 \pm 1.06}$	$\mathbf{82.12 \pm 1.06}$	$\mathbf{82.20 \pm 1.06}$
satimage	78.55 ± 1.80	78.00 ± 1.81	78.00 ± 1.81	78.45 ± 1.80	77.39 ± 1.83	77.39 ± 1.83	77.05 ± 1.83	77.25 ± 1.83
chess	89.06 ± 1.87	88.03 ± 1.94	88.13 ± 1.93	88.03 ± 1.94	88.50 ± 1.91	88.50 ± 1.91	88.50 ± 1.91	88.41 ± 1.91
vehicle	$\mathbf{67.69 \pm 1.61}$	62.60 ± 1.67	63.07 ± 1.66	62.84 ± 1.66	$\mathbf{67.57 \pm 1.61}$	$\mathbf{67.93 \pm 1.61}$	$\mathbf{67.46 \pm 1.61}$	$\mathbf{68.17 \pm 1.60}$
diabetes	77.91 ± 1.50	77.91 ± 1.50	76.99 ± 1.52	76.99 ± 1.52	77.65 ± 1.51	77.65 ± 1.51	77.65 ± 1.51	77.65 ± 1.51
soybean-large	61.07 ± 2.06	84.29 ± 1.53	$\mathbf{92.14 \pm 1.14}$	88.39 ± 1.35	72.66 ± 1.88	62.50 ± 2.05	62.32 ± 2.05	62.86 ± 2.04
vote	92.17 ± 1.77	92.61 ± 1.73	95.21 ± 1.41	93.48 ± 1.63	93.48 ± 1.63	93.91 ± 1.58	93.91 ± 1.58	93.91 ± 1.58
heart	85.19 ± 2.16	85.19 ± 2.17	84.07 ± 2.22	84.07 ± 2.22	84.07 ± 2.22	84.07 ± 2.22	84.07 ± 2.22	84.07 ± 2.22
glass	93.81 ± 1.66	88.57 ± 2.20	95.24 ± 1.47	92.38 ± 1.83	92.86 ± 1.78	93.81 ± 1.66	91.90 ± 1.88	91.90 ± 1.88
iris	93.33 ± 2.03	92.00 ± 2.21	92.67 ± 2.12	93.33 ± 2.03	92.67 ± 2.12	93.33 ± 2.03	92.67 ± 2.13	93.33 ± 2.02
lymphography	79.31 ± 3.36	77.93 ± 3.44	77.24 ± 3.48	74.48 ± 3.62	74.48 ± 3.62	74.48 ± 3.62	73.79 ± 3.65	73.10 ± 3.68
hepatitis	$\mathbf{95.00 \pm 2.44}$	$\mathbf{96.25 \pm 2.12}$	$\mathbf{93.75 \pm 2.71}$	$\mathbf{93.75 \pm 2.71}$	86.25 ± 3.85	83.75 ± 4.12	86.25 ± 3.85	85.00 ± 3.99

Table 2: Experimental results for all discussed scores and for datasets in Table 1. Results in bold are significant in the sense that their confidence interval does not intersect the remaining ones. Observe that for large datasets Bayesian scores perform well, whereas for small datasets information-theoretic scores perform better, in particular, NML seems the best.

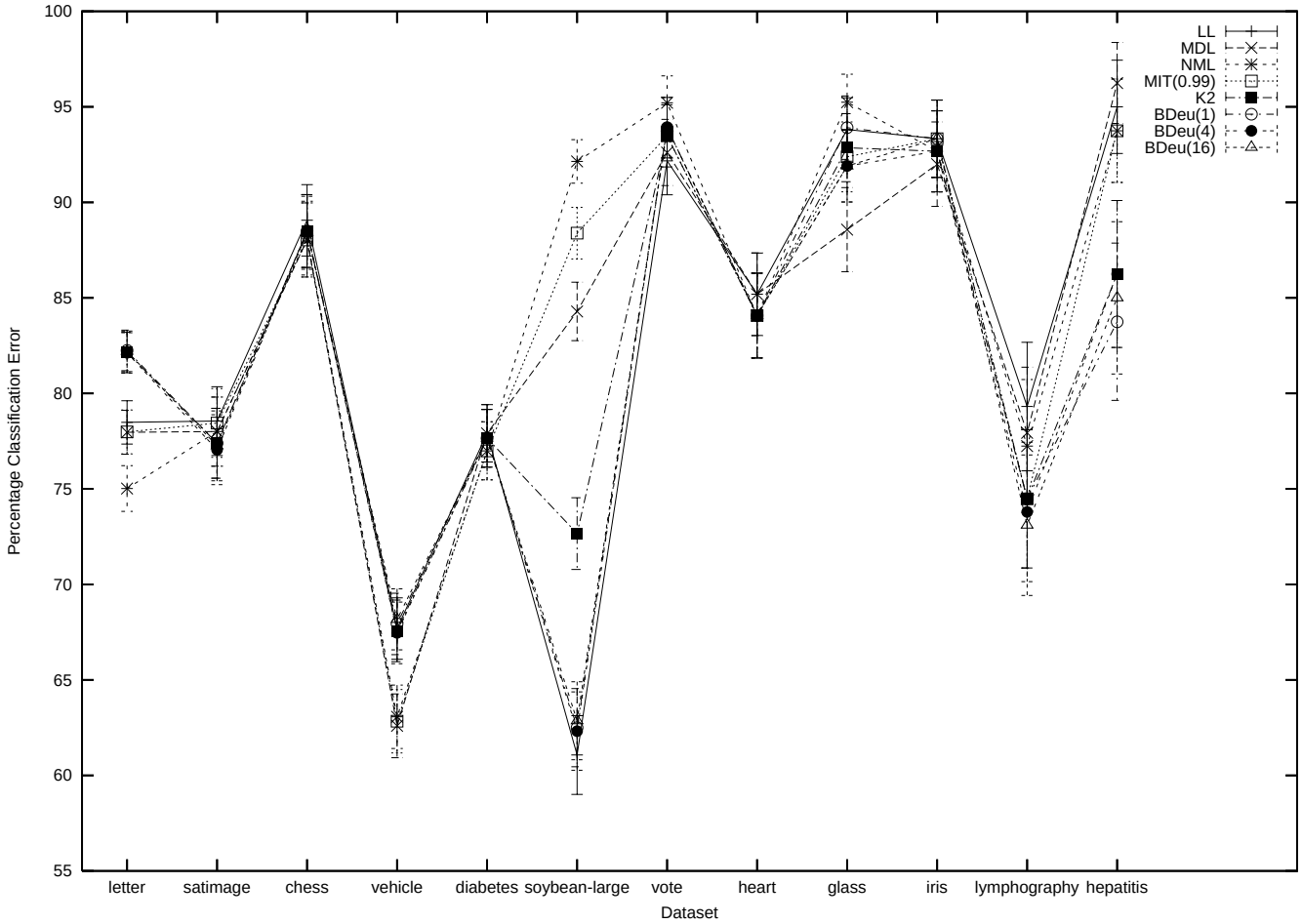

Figure 1: Accuracy curves comparing all discussed scores for datasets in Table 1. The horizontal axis lists the datasets and the vertical axis measures the percentage of test instances that were well classified. Each data point is annotated by a 95% confidence interval. Observe that all metrics perform more or less the same, with the exception of letter, vehicle, soybean-large and hepatitis datasets, where some confidence intervals of different scores do not overlap.

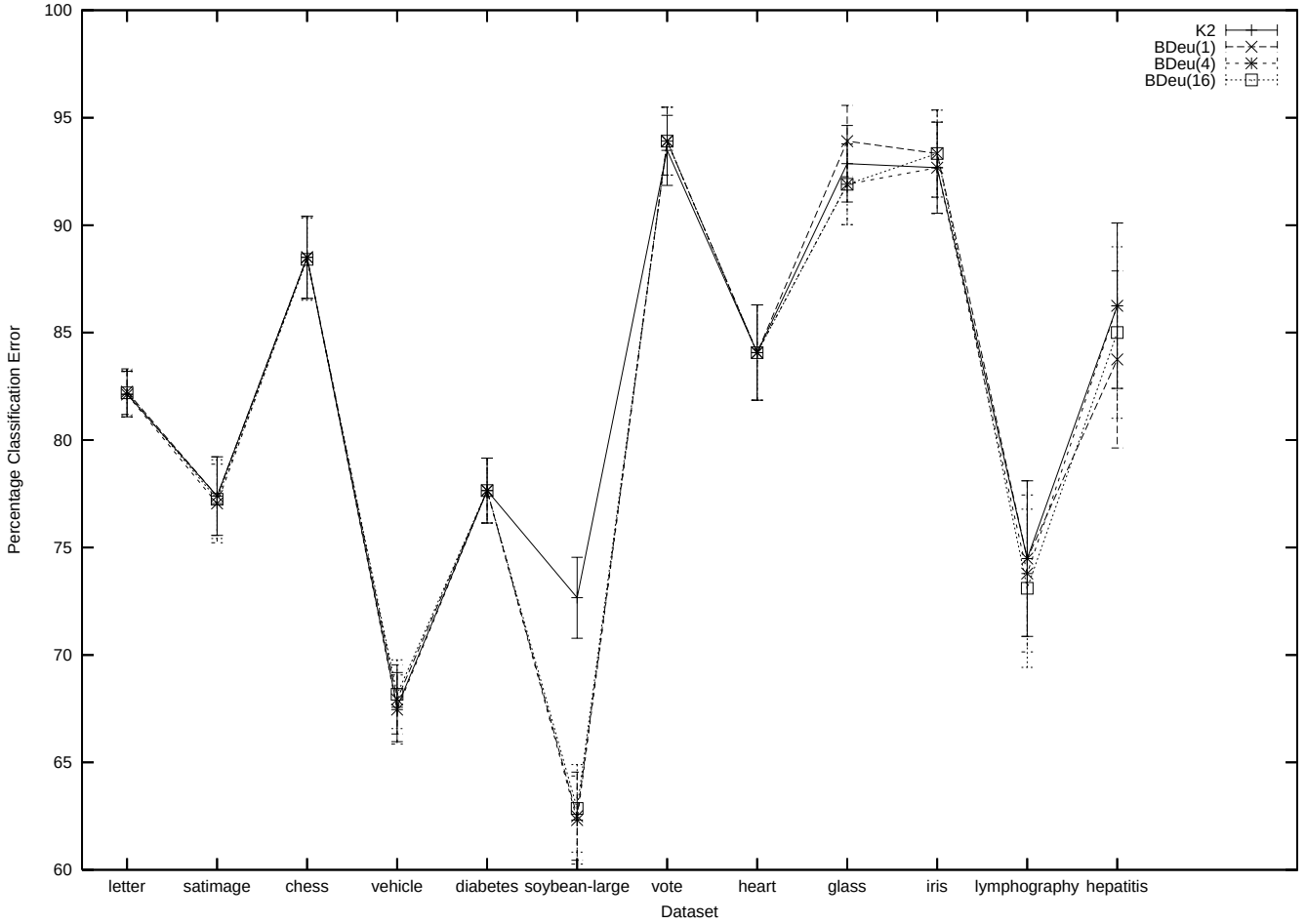

Figure 2: Accuracy curves comparing all discussed Bayesian scores for datasets in Table 1. The horizontal axis lists the datasets and the vertical axis measures the percentage of test instances that were well classified. Each data point is annotated by a 95% confidence interval. Observe that all Bayesian metrics perform more or less the same, with K2 performing slightly better in soybean-large dataset.

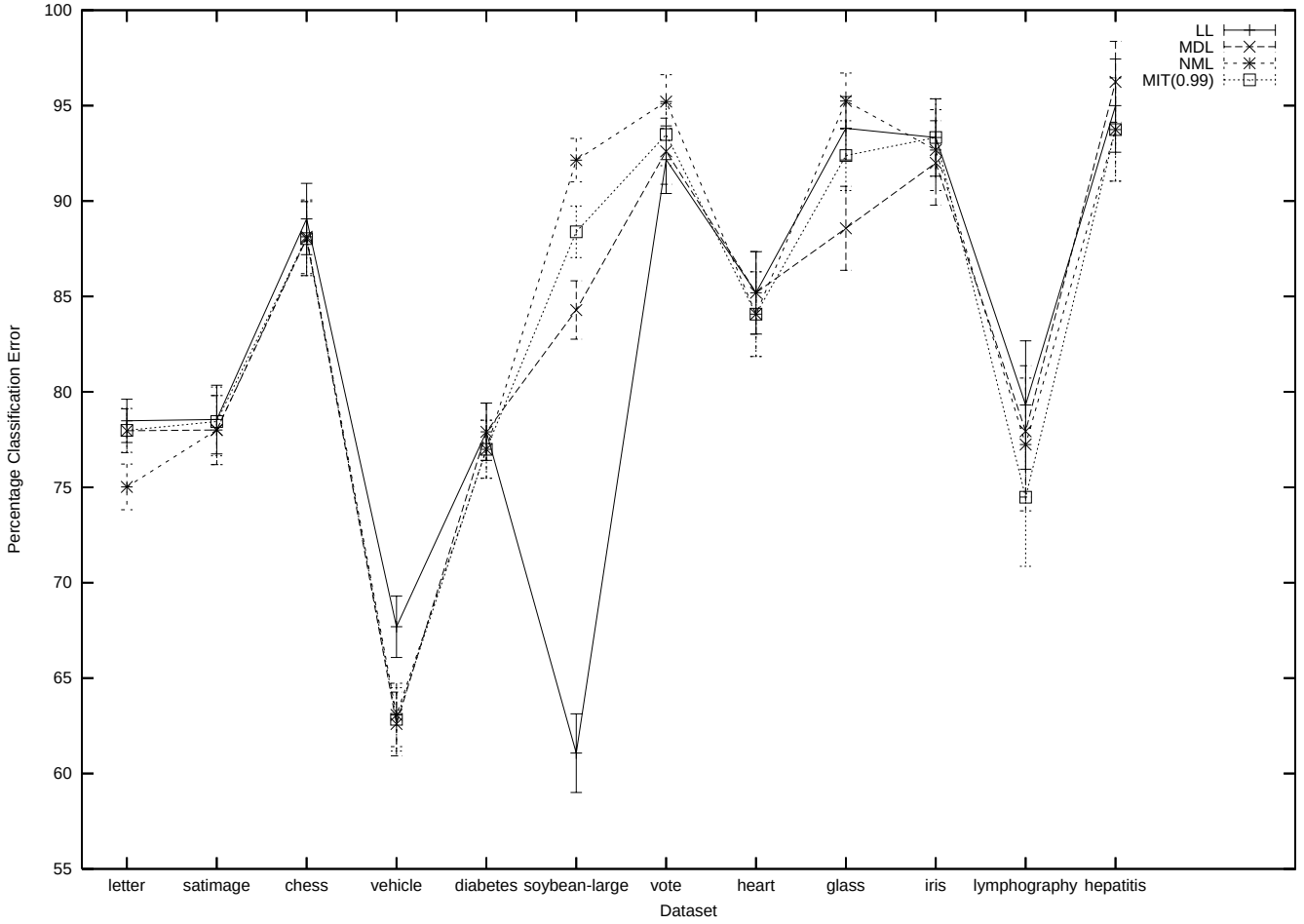

Figure 3: Accuracy curves comparing all discussed information-theoretic scores for datasets in Table 1. The horizontal axis lists the datasets and the vertical axis measures the percentage of test instances that were well classified. Each data point is annotated by a 95% confidence interval. Observe that LL performs well for large datasets whereas the NML score seems the best for small ones.

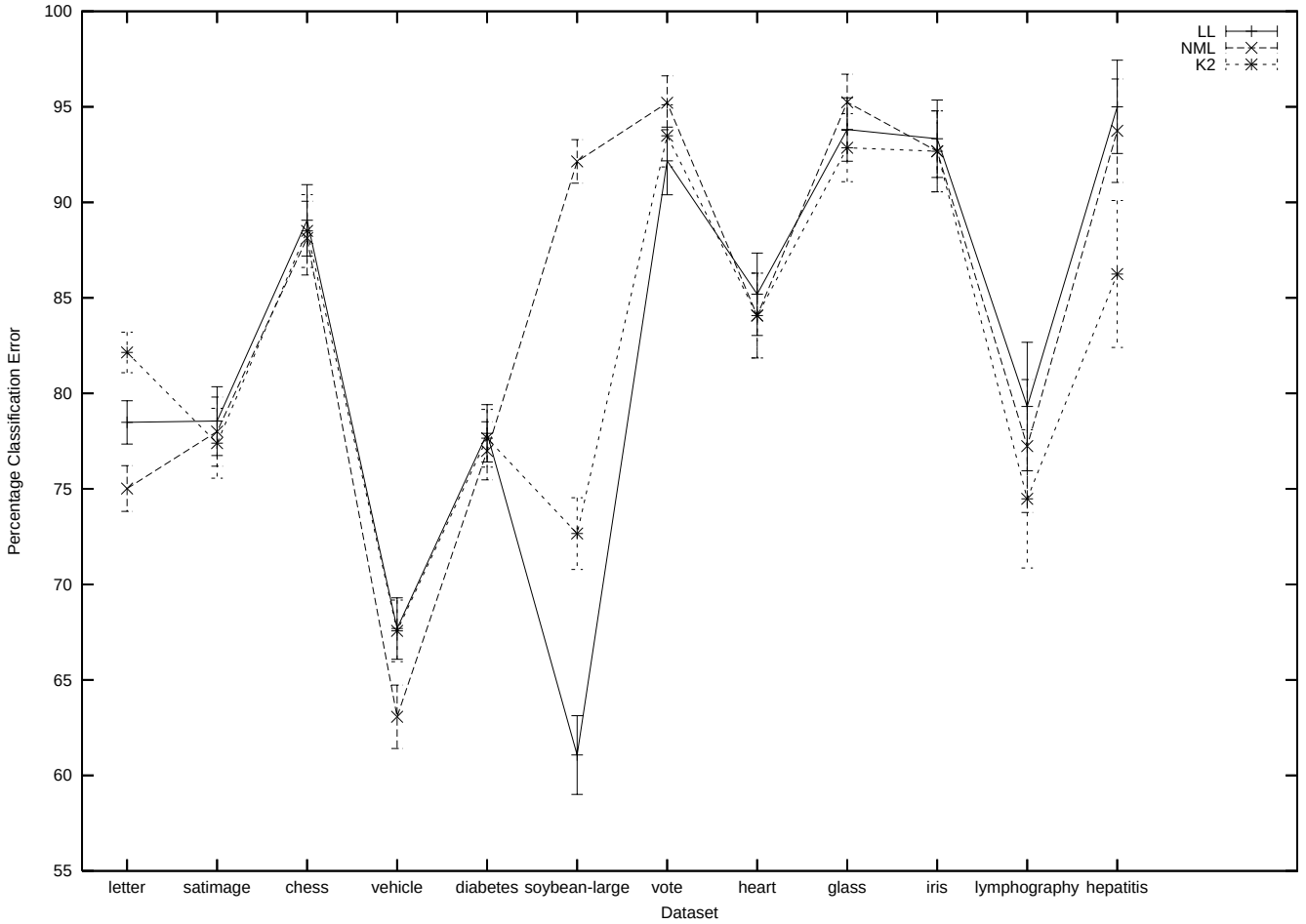

Figure 4: Accuracy curves comparing LL, NML and K2 scores, for datasets in Table 1. The horizontal axis lists the datasets and the vertical axis measures the percentage of test instances that were well classified. Each data point is annotated by a 95% confidence interval. Observe that for large datasets LL and K2 perform more or less the same, with the exception of letter where the K2 is the best. For small datasets NML score is in general the best choice, and LL performs better than K2.

5 Conclusions

The purpose of this work was to benchmark Bayesian network scoring functions for classification. We presented all scores known in the literature together with their justification. We measured the performance of the scores in the task of learning a TAN classifier over UCI datasets. The results show that Bayesian scores are hard to distinguish, performing well for large datasets. The most impressive result was due to the NML score for the soybean-large dataset. It seems that a good choice is to consider K2 for large datasets and NML for small ones.

Future work includes proposing a decomposable score based on conditional likelihood, that will minimize the entropy between the class variable given the values of the attributes, and not the joint entropy of the class variable and the attributes. A version of this score for small datasets endowed with a NML-like penalization should perform very well for biological datasets, such as those used for finding binding sites.

References

[Aka74] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19:716–723, 1974.

[Bou95] R. R. Bouckaert. *Bayesian Belief Networks: From Construction to Inference*. PhD thesis, University of Utrecht, 1995.

[Bun91] W. L. Buntine. Theory refinement on Bayesian networks. In *Proc. UAI'91*, pages 52–60, 1991.

[CH92] G. F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

[Chi96] D. M. Chickering. *Learning Bayesian networks is NP-Complete*, pages 121–130. Learning from data: AI and statistics V. Springer, 1996.

[CL68] C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Trans. Info. Theory*, 14(3):462–467, 1968.

[Coo90] G. F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artif. Intell.*, 42(2-3):393–405, 1990.

[Das99] S. Dasgupta. Learning polytrees. In *Proc. UAI'99*, pages 134–141, 1999.

[dC06] L. M. de Campos. A scoring function for learning Bayesian networks based on mutual information and conditional independence tests. *Journal of Machine Learning Research*, 7:2149–2187, 2006.

[dF37] B. de Finetti. La prévision: See lois logiques, ses sources subjectives. *Annales de l'Institut Henri Poincaré*, 7:1–68, 1937. Translated in Kyburg and Smokler, 1964.

[DH73] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, New York, 1973.

[DL93] P. Dagum and M. Luby. Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artif. Intell.*, 60(1):141–153, 1993.

[Edm67] J. Edmonds. Optimum branchings. *J. Research of the National Bureau of Standards*, 71B:233–240, 1967.

[FGG97] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29(2-3):131–163, 1997.

[FI93] U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proc. IJCAI'93*, pages 1022–1029, 1993.

[GD04] D. Grossman and P. Domingos. Learning Bayesian network classifiers by maximizing conditional likelihood. In *Proc. ICML'04*, pages 46–53. ACM Press, 2004.

[HGC95] D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, 1995.

[KM07] P. Kontkanen and P. Myllymäki. A linear-time algorithm for computing the multinomial stochastic complexity. *Inf. Process. Lett.*, 103(6):227–233, 2007.

[Koh95] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proc. IJCAI'95*, pages 1137–1145, 1995.

[Law76] E. Lawler. *Combinatorial Optimization: Networks and Matroids*. Dover, 1976.

[LB94] W. Lam and F. Bacchus. Learning Bayesian belief networks: An approach based on the MDL principle. *Comp. Intell.*, 10:269–294, 1994.

[NHBM98] D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz. UCI repository of machine learning databases, 1998.

[Pea88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[PS03] Sriram V. Pemmaraju and Steven S. Skiena. *Computational discrete mathematics: combinatorics and graph theory with Mathematica*. Cambridge University Press, 2003.

[Ris86] J. Rissanen. Stochastic complexity and modeling. *Annals of Statistics*, 14:1080–1100, 1986.

[RSKM08] T. Roos, T. Silander, P. Kontkanen, and P. Myllymäki. Bayesian network structure learning using factorized NML universal models. In *Proc. ITA'08*, 2008. To appear.

[RT04] J. Rissanen and I. Tabus. Kolmogorov's structure functions in MDL theory and lossy data compression, 2004.

[Sch78] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

[Sht87] Y. M. Shtarkov. Universal sequential coding of single messages. *(Translated from) Problems of Information Transmission*, 23(3):3–17, 1987.

[Suz93] J. Suzuki. A construction of Bayesian networks from databases based on an MDL principle. In *Proc. UAI'93*, pages 266–273, 1993.

[SZ06] J. Su and H. Zhang. Full Bayesian network classifiers. In *Proc. ICML'06*, pages 897–904. ACM Press, 2006.

[VP90] T. Verma and J. Pearl. Equivalence and synthesis of causal models. In *Proc. UAI'90*, pages 255–270, 1990.

[YC02] S. Yang and K.-C. Chang. Comparison of score metrics for Bayesian network learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 32(3):419–428, 2002.

[ZK04] Y. Zheng and C. K. Kwoh. Improved MDL score for learning of Bayesian networks. In *Proc. AISAT'04*, pages 98–103, 2004.