Progress in Artificial Intelligence (2019) 8:425–439
https://doi.org/10.1007/s13748-019-00194-y

REVIEW

A survey on Bayesian network structure learning from data

Mauro Scanagatta[1] · **Antonio Salmerón**[2] · **Fabio Stella**[3]

Received: 14 March 2019 / Accepted: 20 May 2019 / Published online: 29 May 2019

Abstract
A necessary step in the development of artificial intelligence is to enable a machine to *represent* how the world works, building an internal structure from data. This structure should hold a good trade-off between expressive power and querying efficiency. Bayesian networks have proven to be an effective and versatile tool for the task at hand. They have been applied to modeling knowledge in a variety of fields, ranging from bioinformatics to law, from image processing to economic risk analysis. A crucial aspect is learning the dependency graph of a Bayesian network from data. This task, called *structure learning*, is NP-hard and is the subject of intense, cutting-edge research. In short, it can be thought of as choosing one graph over the many candidates, grounding our reasoning over a collection of samples of the distribution generating the data. The number of possible graphs increases very quickly at the increase in the number of variables. Searching in this space, and selecting a graph over the others, becomes quickly burdensome. In this survey, we review the most relevant structure learning algorithms that have been proposed in the literature. We classify them according to the approach they follow for solving the problem and we also show alternatives for handling missing data and continuous variable. An extensive review of existing software tools is also given.

Keywords Machine learning · Statistics · Bayesian network · Structure learning

1 Introduction

A key issue in artificial intelligence is the development of models able to provide a structured representation of the domain knowledge, and taking into account that most likely those models will be learnt from data. These models should hold a good trade-off between expressive power and querying efficiency. Bayesian networks [50] have proven to be an effective and versatile tool for the task at hand. They have been applied to modeling knowledge in a variety of fields, ranging from bioinformatics to law, from image processing to economic risk analysis [53].

Bayesian networks are a structured knowledge representation, where domain variables are regarded as nodes in a graph whose structure encodes the dependencies between them. A crucial aspect is learning the dependency graph of a Bayesian network from data. This task, which we call *structure learning*, is NP-hard [13], and is the subject of intense, cutting-edge research.

In short, it can be thought of as choosing one graph over the many candidates, grounding our reasoning over a collection of samples of the distribution generating the data. The number of possible graphs dramatically increases with the number of variables. Hence, searching in this space, and selecting a graph over the others, becomes quickly burdensome. In this survey, we present a review of the many sophisticated algorithms that have been proposed in the literature.

The remainder of this paper is organized as follows. Section 2 formally introduces Bayesian networks and gives an illustrative example. Section 3 presents a review of the most relevant literature about structure learning. Section 4 reviews the main software tools implementing Bayesian networks structure learning, and Sect. 5 gives some hints about what

✉ Mauro Scanagatta
mscanagatta@fbk.eu

Antonio Salmerón
antonio.salmeron@ual.es

Fabio Stella
stella@disco.unimib.it

[1] Fondazione Bruno Kessler, Trento, Italy

[2] Department of Mathematics and Center for the Development and Transfer of Mathematical Research to Industry (CDTIME), University of Almería, Almería, Spain

[3] Department of Informatics, Systems and Communication, University of Milan-Bicocca, Milan, Italy

method to use depending on the target application. The paper ends with some concluding remarks in Sect. 6.

2 Bayesian networks

We start with the *earthquake* example given by Pearl [50]. It is a Bayesian network of five variables, modeling the following domain: Mr. K is at work when he receives a call from a neighbor telling him that his house alarm went off. When he is about to call the police, he remembers reading in the instruction manual that the alarm is sensitive to earthquakes and can be accidentally triggered by one. He reckons that if an earthquake occurred, it would be on the news. So he turns on the radio and finds out that an earthquake has just occurred. Should Mr. K call the police? A variable is assigned to each event: the alarm ringing (variable **A**), a burglar intrusion (variable **B**), the earthquake occurrence (variable **E**), the radio announcement (variable **R**), and the neighbor's call (variable **C**). To each variable is then assigned a conditional probability table (CPT). These tables encode the distribution of the variable given the values of its parents. Each variable is binary, with the value t meaning the event has happened, and the value f meaning it has not. The Bayesian network is depicted in Fig. 1.

2.1 Definition

We can now properly define a Bayesian network as a pair $(\mathcal{G}, \mathcal{O})$ that encodes a joint probability distribution over a finite set $\mathcal{X} = \{X_1, \ldots, X_n\}$ of categorical variables. It is composed of: (i) a directed acyclic graph (DAG) $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ whose nodes $\mathcal{V}$ correspond to the variables in $\mathcal{X}$ and arcs $\mathcal{E}$ represent direct dependencies between variables, and (ii) a collection of conditional mass functions $\mathcal{O}$ that define the behavior of each variable X_i given its parents Π_i in the graph.

In the earthquake example, we have $\mathcal{X} = \{A, B, C, R, E\}$; $\mathcal{G}$ is the graph shown in Fig. 1, and $\mathcal{O}$ is the set of conditional probability tables showed in the same figure.

The representation of the full joint table $P(\mathcal{X})$ takes exponential space in the number of variables n. This complexity can be avoided thanks to the so-called Markov condition, which states that in a Bayesian network every variable is conditionally independent of its non-descendant non-parents given its parents. By using this condition, it is possible to represent the joint table in a compact form, as a multiplication of local mass functions:

$$p(\mathbf{x}) = p(x_1, \ldots, x_n) = \prod_i p(x_i|\pi_i),$$

where $\mathbf{x}$ is an instantiation of all the variables in $\mathcal{X}$, x_i is the value of variable X_i in $\mathbf{x}$, π_i is an instantiation of all the variables in the parent set Π_i compatible with $\mathbf{x}$. The representation now takes still exponential space but only in the size of the largest parent set.

In our example, representing the full joint table would take $2^5 = 32$ parameters. Applying the previous decomposition:

$$P(E, B, R, A, C) = P(E)P(B)P(R|E)P(A|E, B)P(C|A), \quad (1)$$

we can represent it with only 20 parameters ($2+2+2^2+2^3+2^2$). The benefit becomes more important as the number of variables increases. (Additionally, it is possible to represent a distribution without the last parameter, as it can be computed from the others. In this case, the full joint table takes 31 parameters, and the decomposition takes 15: 1+1+3+7+3.)

The most important application of a Bayesian network is probabilistic inference, i.e., estimating the posterior probability $P(X|Y)$ on target variables X given evidence on other variables Y. Our goal could be, for example, to decide whether Mr. K has to rush home. This is based on the probability that there has indeed been a burglar intrusion, given

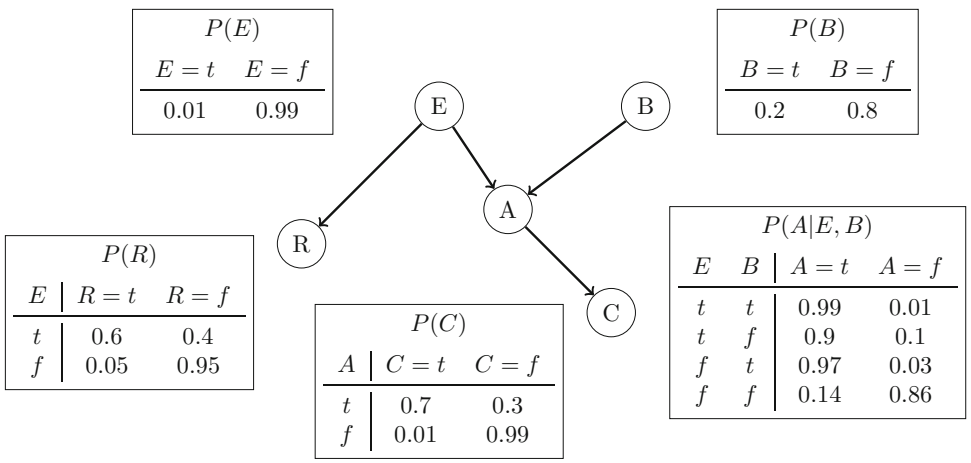

Fig. 1 A Bayesian network for the earthquake problem. The tables specify the probability distribution for each variable conditional on its parent values. All the variables are binary (*true*/*false* values)

Fig. 2 Example data $\mathcal{D}$ sampled from the modeled network

A	B	C	E	R
y	n	n	y	y
n	n	n	n	y
y	y	y	n	n
		...		

that his neighbor called and there was a radio announcement of an earthquake. In a formula:

$$P(B = t|R = t, C = t) = ? \,. \tag{2}$$

Several researchers have studied this application, which has been proven to be NP-Hard [16]. An important result is that this complexity can be managed by bounding the treewidth, a measure of the tree-likeness of the graph. The *treewidth* is defined as the size of the largest clique in a chordal completion of the graph. Modern exact inference algorithms have worst-case time complexity exponential in the treewidth of the underlying graph [39].

3 Structure learning

Suppose now that we are unaware of the original structure of the *earthquake* model. We can only observe its behavior, sampling data from the distribution associated with the model as in Fig. 2. Grounding our reasoning on these observations, we can then make an assessment of their originating structure. This is the fundamental idea of structure learning.

The problem of learning the structure of a Bayesian network from a complete dataset of d datapoints $\mathcal{D} = \{D_1, \ldots, D_d\}$ corresponds to determining the set of directed arcs $\mathcal{E}$ for the DAG $\mathcal{G} = (\mathcal{X}, \mathcal{E})$, using some criterion that specifies the quality of a structure. This can also be stated as choosing for each variable X_i its parent set Π_i.

The usual assumption is for the data to be complete. The case of missing data currently represents a bottleneck for structure learning, as few methods can properly manage it. We discuss them in Sect. 3.3.

The task is computationally non-trivial due to the enormous size of the space of possible graphs $\mathcal{G}$, growing super-exponentially in the number of nodes n [54]. In Fig. 3, we show only a few examples. How can we choose one over the others?

3.1 Score-based structure learning

The most used approach to structure learning is called *score based*. Loosely speaking, the task is to find the best DAG according to some score function that measures its fitness to the data [31]. The goal is now to solve:

$$\underset{G \in \mathcal{G}}{\text{argmax}} \; score(G, \mathcal{D}). \tag{3}$$

One of the most widely adopted scores is the Bayes Dirichlet equivalent uniform (BDeu) [10,17,31], which measures the posterior probability of a chosen DAG given the available data while assuming a uniform prior probability distribution on all the possible networks. Another important one is the Bayesian Information Criterion (BIC) [61], which approximates the BDeu. They share the important property of *decomposability*: The score of a DAG is constituted by the sum of the scores of the subgraphs made by each variable X_i with its parent set Π_i:

$$score(G, \mathcal{D}) = \sum_{X_i} score(X_i, \Pi_i, \mathcal{D}). \tag{4}$$

The idea is now to separate the learning process in two steps: *parent set identification* and *structure optimization*. Parent set identification produces a list of suitable candidate parent sets for each variable, in the order of decreasing score. Structure optimization assigns a parent set to each node, maximizing the score of the resulting structure while avoiding cycles. Most of the approaches focus on the second phase and assume that the complete list of candidate parent sets has been pre-computed.

Structure optimization can further be distinguished between *general structure optimization*, with no constraints on the resulting graph, and *bounded treewidth structure optimization*, with an upper bound for the treewidth of the resulting graph.

Parent set identification Most of the existing approaches to score-based structure learning require a complete exploration

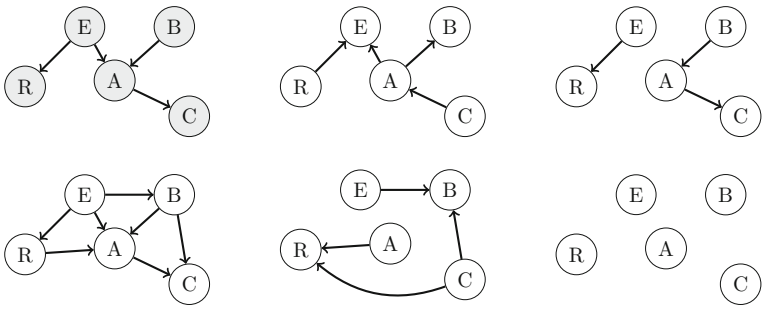

Fig. 3 Possible graph reconstruction of the *earthquake* model. The first graph, with the gray nodes, is the original one

of the space of possible parent sets for every variable, so as to generate a database of pre-computed scores.

The complete exploration of such a space is a burdensome task, as its size grows as $O(n^k)$, where n is the number of variables and k is the maximum number of parents allowed for a node, called *maximum in-degree*. This is a local property in the graph, usually constrained for managing the exploration complexity. Cussens et al. [19] perform structure learning on datasets with 1614 variables, setting $k = 2$, a very limiting constraint. In the case of $k \geq 3$, the largest network studied by Cussens [18] has 60 nodes. These are among the largest examples of score-based structural learning in the literature. Pruning rules have been proposed to help with this matter [21], but they do not considerably reduce the dimension of this space. An exception is the greedy search of the K2 algorithm [17], which has however been superseded in terms of accuracy by the more modern approaches mentioned above.

A high in-degree implies a larger search space and allows achieving a higher score; however, it also requires higher computational time. When choosing the in-degree, the user makes a trade-off between these two objectives. When the number of variables is large, the in-degree is generally set to a small value, in order to allow the optimization to be feasible. The problem of parent set identification is unlikely to admit a polynomial-time algorithm with a quality guarantee [35]. This motivates the development of effective search heuristics.

Scanagatta et al. [60] devise a novel approach for parent set identification that eliminates the limitation posed by the structure constraint k. The idea is to guide the exploration toward the most promising parent sets on the basis of an approximated score function, denoted as BIC^*, that is computed in constant time. The approach is based on the BIC score and is currently the only score function supported. The authors report that BIC^* scales up to thousands of nodes without constraints on the maximum in-degree.

General structure optimization Several exact algorithms for structure optimization have been proposed. The first approaches were based on the observation that for any fixed ordering of the n variables, the decomposability of the score enables efficient optimization over all DAGs compatible with the ordering [17].

Koivisto and Sood [36] and Silander and Myllymaki [63] explored dynamic programming approaches. The main idea is to solve small subproblems first and use the results to find solutions to larger problems until the global learning problem is solved. They spend memory and time proportional to $n \cdot 2^n$. Such complexity limits the realistic application of these approaches only to small domains.

Malone [44], Yuan and Malone [67] and Yuan et al. [68] propose a new approach that formulates the learning goal as a shortest path problem. With the guidance of a consistent heuristic, the algorithm is able to focus on searching for the most promising parts of a solution space. The main benefit consists in the pruning part of the graph space. Empirically, the approach was shown to outperform the competitors on datasets with up to 26 variables.

An approach based on branch and bound was proposed by de Campos et al. [22] and de Campos and Ji [21]. It first finds optimal parent sets for the individual variables by ignoring the acyclicity constraint and then detects and breaks all the cycles to find a valid Bayesian network. It continues to improve on it either by finding a better solution or by reducing the upper bound, until the optimal solution is found. This anytime behavior is the main appeal of the approach, as it allows the user to trade-off between the score of the learned graph and required computational time. It is able to handle datasets with up to 100 variables.

Yet, the current state-of-the-art approach is based on integer linear programming [18,32]. The learning is posed as a relaxed integer linear problem, where cutting planes are introduced for breaking cycles until the optimal solution is found. Identifying good cutting planes is the key to the success of this approach. Experimentally, it has been shown to handle datasets with up to 200 variables.

Malone et al. [43] propose a supervised solver selection. They evaluate a collection of features computed on the input data in order to predict the best state-of-the-art solver for the particular instance. Their work is a promising initial step also in the understanding of each solver weaknesses and strengths.

An interesting idea was introduced by Teyssier and Koller [66] (which we call ordering-based search or *OBS*), where a simple, yet effective heuristic approach is proposed. It is based on a Monte Carlo search over the space of variable orderings, trying to select for each ordering the best network consistent with it. As an approximate method, it is competitive when the problem is too complex for the exact approaches, potentially scaling to datasets with thousands of variables.

OBS employs a naïve operator, called *swap*, for escaping local maxima in the local greedy search. A more powerful operator, called *insertion*, was proposed by Alonso-Barba et al. [4]. With this operator, the authors were able to significantly increase the score of the learned structures.

Recently, Lee and van Beek [41] suggest enhancing the performance of the greedy local search by applying a metaheuristic. They propose two approaches: (a) an iterated local search coupled with a perturbation factor and (b) a genetic algorithm in which each candidate solution is an individual, subject to a series of crossover and mutation stages.

Scanagatta et al. [58] propose a novel *operator* for improving the score of the structure by applying local changes to its underlying ordering. They denote it as the *window* operator for local search, already known as block insertion [7] in the local search literature. They report that the resulting algorithm consistently yields higher-scoring networks than

state-of-the-art competitors, and is able to learn Bayesian networks from dataset containing thousands of variables.

The idea to manipulate the underlying ordering was also explored by Scanagatta, Corani, de Campos and [57]. The authors investigate which parent sets are especially important for the score of a DAG, with the aim of improving the sampling strategy and thus to cover better regions of the space of orders. Their approach is to sample the variables of the ordering in a fashion that is proportional to their entropy. This leads to an increase in performance that can be applied to any algorithm based on local search over the space of orderings.

Local search has also been carried out over the space of equivalence classes of network structures [3,12] resulting in the so-called GES algorithm. It has recently been improved by introducing new local search heuristics, scaling up to networks of 100 nodes [5].

Zheng et al. [69] have recently proposed a novel and interesting approach to structure learning. They have been able to formulate the learning task as a continuous optimization problem that can be solved by standard numerical algorithms.

Bounded treewidth structure optimization Being a global property of the graph, bounding the treewidth considerably increases the difficulty of the learning process. The treewidth of a DAG is measured as the treewidth of its *moral graph*, that is, the undirected graph obtained by adding an undirected edge between all the common parents of a node, and discarding the direction of all the other arcs.

The first approaches were based on heuristic edge additions. An approximate method has been proposed by Elidan and Gould [23], polynomial in both the number of variables and the treewidth bound. It is able to provide an upper bound on the treewidth of the learned structure, thus able to check at each arc addition if the bound is violated. The experiments show that their approach is able to learn better models than competing heuristic methods, on datasets with up to 200 variables.

An exact method has been proposed by Korhonen and Parviainen [38]. It is a dynamic programming algorithm able to find the optimal graph maximizing a given score function in time $O(3^n n^{k+O(1)})$, where k is the treewidth bound. Experiments show its applicability to networks with up to 15 nodes and treewidth bound of 3. Its complexity growth may render it unfeasible for larger domains.

Parviainen et al. [49] adopted an integer programming approach. Cutting planes are formulated in order to enforce the treewidth bound. Their experiments showed an improvement over the competitors on datasets with up to 60 nodes.

Nie et al. [47] proposed an approximation technique that consists of sampling k-trees (maximal graphs of treewidth k), and subsequently selecting the best structure which is a subgraph of that k-tree. Their experiments showed promising results for datasets with up to 100 variables. Unfortunately, the sampling space for k-trees is huge and can barely be effectively explored by the sampling technique in a reasonable amount of time. As the number of variables grows, each sampled k-tree has a low probability of providing a good fit for the given domain, and the time required to find good solution increases.

Nie et al. [46] improved on the above idea by searching for promising k-trees using an A*-guided exploration. The goodness of the k-tree is approximated by using a heuristic evaluation, called *Informative Score*. When a promising k-tree has been selected, the algorithm searches for the optimal DAG that is subgraph of the k-tree. The idea has been shown to improve on competitors, but does not solve completely the issue about the difficulties of sampling in the space of k-trees.

Scanagatta et al. [56] exploit the fact that any k-tree can be constructed by an iterative procedure that adds one variable at a time. They then propose an iterative procedure that, given an order on the variables, builds a DAG $\mathcal{G}$ adding one variable at a time. The moral graph of $\mathcal{G}$ is thus ensured to be subgraph of a k-tree. The algorithm is designed as to maximize the score of the resulting DAG while guaranteeing the bound on the treewidth. They report that their approach scales effectively with both the number of variables and the treewidth, outperforming the competitors, and is able to learn bounded treewidth Bayesian networks from dataset containing thousands of variables.

3.2 Constraint-based structure learning

There are also methods based on statistical tests, called *constraint-based* structure learning. These algorithms use a series of conditional hypothesis tests to learn independences among the variables in the model [51,52]. Following these constraints, the DAG is in turn built. Their performance is critically determined by the adopted hypothesis test.

The state-of-the-art approach is the PC algorithm (named after its authors, Peter and Clark) introduced by Spirtes et al. [64]. It starts from a complete, undirected graph and deletes edges recursively based on conditional independence decisions. This yields an undirected graph that can then be partially directed and further extended to represent the underlying DAG. The PC algorithm runs in the worst case in exponential time (as a function of the number of nodes), but if the true underlying DAG is sparse, which is often a reasonable assumption, this reduces to a polynomial runtime. Kalisch and Bühlmann [34] showed the applicability of the PC algorithm even in high-dimensional settings, up to 1000 nodes.

The standard PC algorithm relies on an ordering over the variables, and may thus suffer from unstable results. Colombo and Maathuis [14] proposed a modification of the

algorithm that significantly increases the robustness of the method, rendering it order independent.

The PC algorithm by itself does not require a specific conditional independence test. Most applications adopt Chi-square test or mutual information tests. An interesting approach, based on Bayesian score tests, has been proposed by Abellán et al. [1].

Recent innovations introduced parallel computing techniques for local causal discovery methods in constraint-based approaches [62]. Remarkably, the solution proposed by Madsen et al. [42], which is based on allocating the conditional independence tests in the available computing units following a balanced incomplete block design, has been tested on datasets of over 2000 variables and 500k records.

3.3 Incomplete data

The standard approach to structure learning requires the data to be complete. At the moment, very few methods can handle the case of incomplete data. The most commonly used is the structural expectation–maximization (SEM) [27]. This approach performs a two-step iterative procedure: (i) *E-step*: Complete the data with expected values or modes from the currently induced Bayesian network (in other words, compute the expected sufficient statistics); (ii) *M-step*: Perform structure learning as if the data were complete. This procedure continues until convergence. Unfortunately, the complexity cost of this computation can be very high and restrict its application only to small domains.

A competing approach has been proposed by Adel and de Campos [2]. Given the task of learning a Bayesian network from missing data, it translates it into the problem of structure learning from complete data. This is done by augmenting the network with variables that take into account all the possible completions of the data. The approach showed convincing potential in datasets with up to 23 variables, but only on small datasets, since it has an exponential cost in the size of the missing data.

A different approach was proposed by Fernández et al. [24]. It is designed for learning Bayesian networks representing regression problems, and is based on using mixtures of truncated exponentials in order to represent the joint distribution of the induced network [45]. It relies on a data augmentation algorithm that iteratively re-estimates a Bayesian network model, and inputs the missing values using it. It showed promising results on datasets with up to 16 variables, with both discrete and continuous values.

Nielsen et al. [48] proposed an algorithm for learning, in the presence of missing data, a Probabilistic Decision Graph: This is a representation language for probability distributions based on binary decision diagrams. It can encode independence relations that cannot be captured in a Bayesian network structure, and can sometimes provide computationally more efficient representations than Bayesian networks [33]. Their experiments showed a reasonable performance, but on networks with only up to 40 variables.

Scanagatta, Corani, Zaffalon, Yoo and Kang [59] propose a method for performing structure learning on incomplete datasets, using structural expectation–maximization (SEM). They tackle the issue of the complexity cost by employing a bounded treewidth Bayesian network structure learning algorithm. In this way, they obtain a fast implementation of SEM, since the bounded treewidth structures learned in the different iterations guarantee efficient inferences. They report that it is the first implementation of SEM that is able to scale thousands of variables.

3.4 Handling continuous variables

The methods for structure learning described so far are basically designed for discrete variables, though in general they can be adapted to handle continuous variables as well. The most straightforward approach is to discretize the continuous variables, so that statistical tests of independence can be carried out, and therefore, algorithms such as the PC can be applied. This approach has been explored by Fernández et al. [25].

Score-based structure learning can also be used with continuous variables as long as the score used to guide the search process is compatible with the model adopted for representing the probability density of the continuous variables. The method proposed by Romero et al. [55] represents such densities using mixtures of truncated exponentials.

However, some statistical models are incompatible with this approach, as they impose some structure restrictions. This is the case of the conditional Gaussian (CG) model [40], where discrete variables are not allowed to have continuous parents. This model corresponds to scenarios where the marginal over the discrete variables is a Multinomial and the conditional density of the continuous variables for each configuration of the discrete ones is a multivariate Normal. A solution for learning CG networks following a score-based approach was proposed by Bøttcher [8], later improved by considering equivalence classes resulting in a more efficient search procedure [9].

3.5 Statistical classification

Statistical classification is a common problem in machine learning, where it is required to identify to which category a new observation belongs, based on the previous observations. The category to be predicted is known as the *class*, and the explanatory variables on which the choice is made are called *features*. An example would be to predict whether tomorrow is going to rain (*class*) given information on today's atmospheric conditions such as pressure, humidity, wind

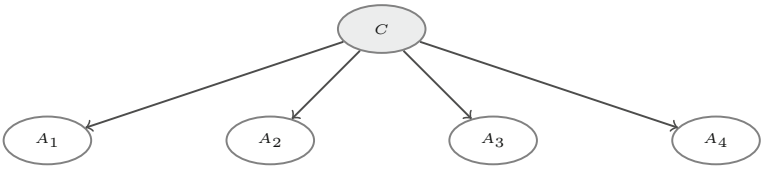

Fig. 4 Naïve Bayes. In this figure and the following ones, class variable is shown in gray

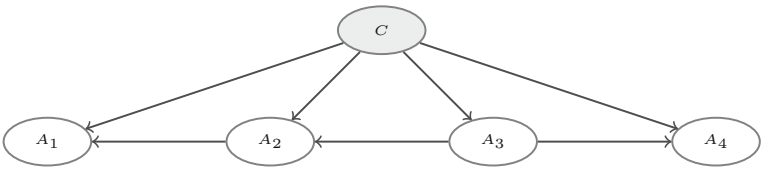

Fig. 5 TAN: The tree is $A_1 \leftarrow A_2 \leftarrow A_3 \rightarrow A_4$

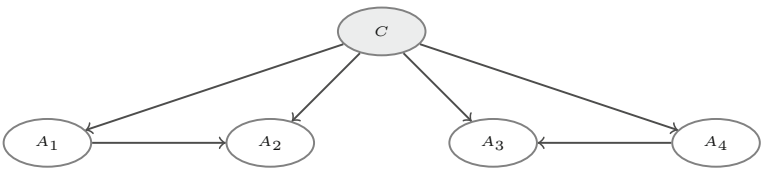

Fig. 6 FAN: The forest is constituted by the sub-trees $A_1 \rightarrow A_2$ and $A_3 \leftarrow A_4$

speed (*features*). This has been a widely used application of Bayesian networks. Thanks to structural constraints, the approaches are usually highly scalable, able to learn optimal structures even with thousands of variables.

Naïve Bayes (NB) has been one of the first proposed tools. An example is shown in Fig. 4. It assumes the stochastic independence of the features given the class. This strong assumption makes the classifier highly scalable, with a complexity of $O(n)$. However, the posterior probabilities are biased and may lead to incorrect estimation [29].

The tree-augmented naïve classifier (TAN) [28] relaxes this assumption, augmenting the NB with a tree which connects the features. An example is shown in Fig. 5. As a result, one feature has only the class as a parent, while the remaining features have two parents: the class and another feature. This design consistently helps reducing the bias of naïve Bayes, at the cost of time complexity of only $O(n^2)$ for exact learning.

TAN is further improved by the forest-augmented naïve classifier (FAN). An example is shown in Fig. 6. A FAN augments the naïve Bayes with a forest. A forest is a set of disjoint trees; it is more general than a tree, as it includes the tree as a special case. Thus, the BIC score of FAN is higher than or equal to the BIC score of TAN. The structural learning algorithm of FAN [37] is obtained as a slight modification of the TAN algorithm. It maintains the same quadratic time complexity.

A limit of both TAN and FAN is that they do not perform feature selection; each feature is forcedly connected to the class without checking whether it is relevant. The extended tree-augmented naive (ETAN) [20] overcomes this problem. ETAN allows each feature to have as parent either (i) the class; (ii) the class and a feature; (iii) a feature without the class; (iv) no parent, in which case the feature is recognized as irrelevant. The structural learning algorithm of ETAN exactly identifies the highest-scoring graph that satisfies the previous constraints. An example is shown in Fig. 7. This algorithm is more complex than that of TAN and FAN. The ETAN includes naive Bayes, TAN and FAN as special cases; thus, it achieves a higher BIC score (equal score in the worst case) than all of them.

Bayesian network classifiers have also been extended to regression problems, in which the variable to predict is continuous. Adaptations for learning different structures in regression problems have been proposed by Fernández and Salmerón [26], where feature selection is carried out following a filter approach.

4 Software tools

This section provides the current state of the art on software tools for learning Bayesian networks from data. A summary

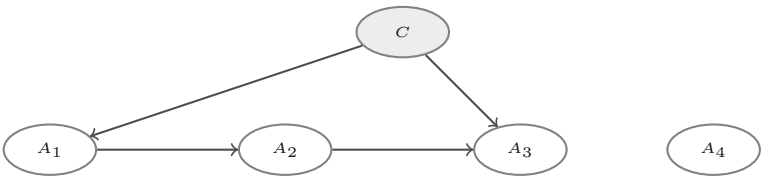

Fig. 7 ETAN which recognizes A_4 as an irrelevant feature. Note also that A_2 is not connected to the class

table, containing main characteristics of analyzed software, is made available.

4.1 BANJO

Bayesian network inference with Java objects (BANJO) https://users.cs.duke.edu/~amink/software/banjo/ is a software application and framework for structure learning of static and dynamic Bayesian networks, developed under the direction of Alexander J. Hartemink in the Department of Computer Science at Duke University. BANJO focuses on score-based structure learning by making available the following heuristic search strategies: simulated annealing and greedy hill climbing. These heuristics are paired with evaluation of a single random local move or all local moves at each step. BANJO runs on computers running Macintosh OS X, Microsoft Windows, Linux, and Solaris, and its code is modifiable within any Java development environment, from simple text editor to full-featured IDE, while the open-source Eclipse IDE has been used to develop it.

4.2 BayesiaLab

BayesiaLab (http://www.bayesia.com/) is a commercial desktop application (Windows/Mac/Unix) with a sophisticated graphical user interface, which provides scientists a comprehensive "laboratory" environment for machine learning, knowledge modeling, diagnosis, analysis, simulation, and optimization. According to https://library.bayesia.com/display/FAQ/Score-Based+Learning+Algorithms, BayesiaLab performs structural learning by using a score-based approach which optimizes the Minimum Description Length score function. BayesiaLab offers different algorithms in order to increase the probability to find the optimal solution. They are using different search policies (greedy, taboo) and/or are exploring different spaces (Trees, Bayesian networks, Essential Graphs, Order of nodes).

4.3 Bayesian network classifiers in Weka

Bayesian Network Classifiers in Weka (https://www.cs.waikato.ac.nz/~remco/weka_bn/) is a Java source code available developed by the University of Waikato and made available in 2006 under GPL1 license. Structure learning of Bayesian networks is achieved by score-based approach, hill climbing (K2, B, etc.) and general purpose (simulated annealing and tabu search) algorithms. In particular, the local score metrics implemented are: Bayes, BDe, MDL, entropy, AIC. Bayesian Network Classifiers in Weka comes with an easy-to-use graphical user interface. Furthermore, structure learning is made possible also according to the constraint-based approach through the ICS and CI algorithms.

4.4 Bayes server

Bayes Server (https://www.bayesserver.com/) is a commercial tool for modeling Bayesian networks, Dynamic Bayesian networks and Decision graphs. It comes with an effective graphical user interface, and a number of advanced APIs, including all the features of the graphical user interface, are made available. Structural learning of Bayesian networks can be performed according to the score-based approach, and to the constraint-based approach through the PC algorithm. Bayes Server API can be used from C#, Java, JavaScript, Python, R, MATLAB, Excel functions and Apache Spark.

4.5 B-course

B-Course is a web-based data analysis tool for Bayesian modeling which has been made available from November 2002 by the Complex Systems Computation Group CoSCo, at Helsinki Institute for Information Technology. According to the web site http://b-course.hiit.fi/obc/, B-Course allows the user to upload his/her dataset to learn the structure of a Bayesian network by using a score-based approach which exploits proprietary algorithms, as well as simulated annealing and hill climbing algorithms. In particular, the B-Course's proprietary algorithm picks models that resemble the current best model. Indeed, the web site reads "*These models are more likely to be predictively more accurate than the current best model than if we picked the models randomly*".

4.6 BKD/BD

Bayesware Discoverer (https://open.bu.edu/handle/2144/1288) originates from Bayesian Knowledge Discoverer (BKD), i.e., a software tool that can learn Bayesian networks from data by using the "bound and collapse" approach based on probability intervals. The dataset to learn from may contain missing values. BKD is a free software, but it has been succeeded by a commercial version, the Bayesware Discoverer, which offers a nice graphical user interface with some powerful visualization options. However, Bayesware Discoverer is only available for Windows.

4.7 blip

blip (Bayesian network Learning Improved Project) is an open-source Java package that offers a wide range of structure learning algorithms (https://github.com/mauro-idsia/blip). It is developed my Mauro Scanagatta, and it is distributed under the LGPL-3 by IDSIA. It focuses on score-

based learning, mainly the BIC and the BDeu score functions, and allows the user to learn BNs from datasets containing thousands of variables. It provides state-of-the-art algorithms for the following tasks: parent set identification (BIC^*), general structure optimization (WINASOBS-ENT), bounded treewidth structure optimization (KMAX) and structure learning on incomplete datasets (SEM-KMAX). An R binding is also available.

4.8 bnlearn

Bayesian network structure learning, parameter learning and inference is an R package which offers a rich set of algorithms which was first released in 2007 by Marco Scutari. Bnlearn (http://www.bnlearn.com/) allows constraint-based structural learning via the following algorithms: PC, Grow shrink Markov blanket, incremental association based on Markov blanket, inter-incremental association based on Markov blanket, fast incremental association based on Markov blanket, max–min parents and children, Hiton parents and children. Bnlearn allows score-based structural learning through the following algorithms: hill climbing, tabu search, hybrid max–min hill climbing, Restricted Maximization. Some of the many available score functions for discrete variables are: log-likelihood, BDe, BIC, MDL, BDs, K2, while for continuous variable the available score functions are: log-likelihood, AIC, BIC, for Gaussian networks. Furthermore, bnlearn allows for black and white lists.

4.9 BNJ

Bayesian Network tools in Java (BNJ) is an open-source suite of software tools for research and development using graphical models. It is implemented in Java and distributed under the GNU General Public License (GPL) by the Kansas State University Laboratory for Knowledge Discovery in Databases (KDD). It can be downloaded from http://bnj.sourceforge.net/, and the latest release is July 21, 2004. Structure learning is made available through score-based approach, while search strategy includes sparse candidate, Markov chain Monte Carlo (perturbation-based). BNJ works for both Windows and Linux.

4.10 bnstruct

Bnstruct is an R package which learns Bayesian networks from data with missing values (https://cran.r-project.org/web/packages/bnstruct/index.html). It implements the Silander–Myllymaki complete search, the max–min parents and children, the hill climbing, the max–min hill climbing heuristic searches, and the structural expectation–maximization algorithm. Structure learning is score based on the following scoring functions: BDeu, AIC, BIC. The current release is bnstruct 1.0.4, which has been released on August 31, 2018.

4.11 BNT

The Bayes Net Toolbox (BNT) has been developed by Kevin Murphy from 1997 to 2002 using the MATLAB programming language. It offers no graphical user interface, and its last update dates back to October 19, 2007. Starting from January 2014, it is maintained on Github at https://github.com/bayesnet/bnt. BNT supports structure learning of Bayesian networks by making available the following score-based algorithms: Bayesian learning, using MCMC or local search, only in the case where nodes are discrete and fully observed, and the following constraint-based algorithms: Inductive Causation (IC), Fast Causal Inference (FCI) [52], and PC [64].

4.12 DEAL

DEAL is a software package for use with R (https://www.jstatsoft.org/article/view/v008i20) which has been developed by Susanne G. Bøttcher and Claus Dethlefsen. DEAL includes several methods for analyzing data using Bayesian networks with variables of discrete and/or continuous types but restricted to conditionally Gaussian networks. Structure learning is implemented through a score-based approach which uses the Bayes factor comparing two different DAGs. The search strategy compares models that differ only by a single arrow, either added, removed or reversed. Indeed, in these cases, the Bayes factor is especially simple, because of decomposability of the network score. DEAL also offers the possibility to combine the search algorithm random restarts [31], and to exploit equivalence relations to speed up model search. It is worthwhile to mention that DEAL has an interface to Hugin.

4.13 ELVIRA

The ELVIRA system (http://leo.ugr.es/elvira/) is a tool to construct model-based decision support systems [15]. It offers an easy-to-use graphical user interface (GUI) which allows the user to define models by clicking his/her mouse. It has been developed using the Java language, and thus, it can be used on any operating system. Furthermore, the source code is made freely available to download. However, its last update dates back to June 19, 2001, and no information is provided about the implemented structure learning approach.

4.14 Free BN

Free BN is an open-source, Java written, Bayesian network structure learning API licensed under the Apache 2.0 license

which has been made available by Jee Vang as a major accomplishment of his PhD dissertation. Structural learning of Bayesian network is performed according to the score-based approach, using a search-and-score algorithm which optimizes the K2 metric, and according to the constraint-based approach mainly through the PC algorithm. The last interactions on the official web site date back to 2010.

4.15 GeNie and smile

GeNie and Smile are commercial software from BayesFusion (https://www.bayesfusion.com/). GeNie Modeler is a graphical user interface (GUI) to SMILE Engine and allows for interactive model building and learning. It is written for the Windows environment but can be also used on macOS and Linux under Wine. The SMILE Engine offers easy integration of BayesFusion in customers' applications, which can be written in a variety of programming languages (e.g., C++, Python, Java, R, .NET). SMILE.COM allows for easy integration with MS Excel. Structure learning can be performed by the score-based approach algorithms such as: Bayesian search algorithm [31], Greedy Thick Thinning [11] and by using the PC constraint-based algorithm [64]. Furthermore, GeNie makes available the Essential Graph Search algorithm, based on a combination of the constraint-based search (with its prominent representative being the PC algorithm) and the Bayesian Search approach.

4.16 GOBNILP

Globally Optimal Bayesian Network learning using Integer Linear Programming (GOBNILP) is a C program which learns Bayesian networks from complete discrete data or from local scores. GOBNILP (https://www.cs.york.ac.uk/aig/sw/gobnilp/) uses the SCIP (https://scip.zib.de/) framework for constraint integer programming. A development version of GOBNILP is freely available as well as a Python version. Structural learning in GOBNILP is score based (BDeu), while the cutting plane algorithm is used to solve an integer linear programming problem for score optimization. The latest version, GOBNILP 1.6.3 has been released on January 5, 2018, and it is available for both Windows and Mac.

4.17 Hugin expert

Hugin Expert (https://www.hugin.com/) is a commercial software based on Bayesian networks and influence diagram technology. The following products are made available: Hugin Explorer, Hugin Developer, Hugin Educational, which is free but limited in the maximum number of nodes, and Hugin Researcher which makes available API to develop industrial applications. It is Java based, offers an intuitive ed effective graphical user interface and it is available on Windows, Linux and macOS. Structural learning is performed by the score-based approach which allows choosing whether to optimize BIC or AIC, and by the constraint-based approach, where the PC [64] and Necessary Path Condition [65] learning algorithms are made available.

4.18 LibB

LibB (http://www.cs.huji.ac.il/labs/compbio/LibB/programs.html) has been developed by Nir Friedman and Gal Elidan, and it consists of a set of executables freely available for academic use only. It is made available for the following software platforms: Windows 2000, NT, XP, and Linux. The program *LearnBayes* learns the structure of a Bayesian network by the score-based approach. In particular, the available scores are the following: BIC, likelihood, BDe and BDe with equivalent sample size, while the following search options are available: greedy hill climbing, greedy hill climbing with restarts, tabu search, best-first search, beam search, MCMC search, simulated annealing, and stochastic first-ascent search. It is worthwhile to mention that structure learning is made possible also in the case of missing data; in such a case, Structural Expectation Maximization is used.

4.19 Libpgm

Libpgm (https://pythonhosted.org/libpgm/) is an endeavor to make Bayesian probability graphs easy to use. It was developed at CyberPoint Labs during the Summer of 2012 by Charles Cabot working under the direction of James Ulrich and Mark Raugas. It is provided free of use in accordance with the New BSD License. Structure learning can be performed according to both the score-based and the constraint-based approaches.

4.20 OpenMarkov

OpenMarkov (http://www.openmarkov.org/index.php?lang=en) has been developed in Java by the Research Centre for Intelligent Decision Support Systems of the UNED in Madrid. The latest stable release is 0.2.0. Structure learning is performed according to the score-based approach by the hill climbing search procedure applied to the $K2$ metric, and according to the constraint-based approach applying the PC [64] algorithm. OpenMarkov makes available a rudimentary treatment of dataset with missing values, i.e., record removal of missing replacement with the string *missing*. Furthermore, OpenMarkov can be used as an API.

4.21 PNL and PyPNL

The Open-Source Probabilistic Networks Library (PNL) is a tool, written in C++, for working with graphical models made available under the Intel Open-Source License. It supports directed and undirected models, discrete and continuous variables, and various inference and learning algorithms. The last update of PNL (https://sourceforge.net/projects/openpnl/) dates back to April 16, 2013. However, it is made available in Python (https://github.com/PyOpenPNL) with the last update dating back to April 15, 2017. Structure learning is score based, where BIC and AIC scores can be optimized. It is worthwhile to mention that a parallel implementation of PNL is made available from the Information Technologies Laboratory, Faculty of Computational Mathematics and Cybernetics at Nizhni Novgorod State University.

4.22 Pomegranate

Pomegranate is a Python package that implements fast and flexible probabilistic models ranging from individual probability distributions to compositional models such as Bayesian networks and hidden Markov models (https://pomegranate.readthedocs.io/en/latest/BayesianNetwork.html). Structure learning of Bayesian networks from data is performed according to the score-based approach. However, the search strategy is such that it currently enumerates all the exponential number of structures and finds the best according to the score. The score that is optimized is the minimum description length (MDL).

4.23 ProBT

ProBT (http://www.probayes.com/en/recherche/probt/) is commercial C++ library which makes available advanced modeling, inference, and learning capabilities. It extends the Bayesian networks framework by providing a structured programming language allowing the developers to increase their applications capabilities and robustness by integrating Bayesian models. Structure learning is made possible by score-based approach by optimizing MI, BIC, MDL, and AIC scores.

4.24 Tetrad

The Tetrad codebase (http://www.phil.cmu.edu/tetrad/), developed in JAVA, and copyrighted by Clark Glymour, Richard Scheines, Peter Spirtes and Joseph Ramsey, is publically available on GitHub (https://github.com/cmu-phil/tetrad). Tetrad is a unique suite for causal discovery; it provides algorithms the capability to discover causal models when there may be unobserved confounders of measured variables, to search for models of latent structure, and to search for linear feedback models. Tetrad V comes with a easy-to-use graphical user interface, and it allows calculating predictions of the effects of interventions or experiments based on a model. Many algorithms for learning the structure of Bayesian networks and causal Bayesian networks are made available. Score-based algorithms include: FGES and IMaGES, for both continuous and discrete variables as well as for missing data. Constraint-based algorithms include: PC, PCStable, CPC, CPCStable, PCmax, FCI, RFCI, GFCI, TsFCI, etc. For additional information, the reader is referred to the Tetrad V manual which is available at http://www.phil.cmu.edu/tetrad/.

4.25 UnBBayes

UnBBayes (http://unbbayes.sourceforge.net/index.html) is an open-source software for modeling, learning and reasoning upon probabilistic networks developed in Java by researchers from the University of Brasília and the George Mason University. The web site mentions that structural learning of Bayesian networks is implemented trough the score-based approach, i.e., K2 metric. However, the latest update of UnBBayes dates back to March 24, 2010.

4.26 WinMine/MSBN

The WinMine/MSBN toolkit is free for non-commercial purposes. It has been developed by Max Chickering at Microsoft Research, and works on Windows 2000/NT/XP and can be downloaded at https://www.microsoft.com/en-us/research/project/winmine-toolkit/. It was established on October 5, 2002, and consists of a set of executables which allow to build statistical models from data. Learning the structure of a Bayesian network can be performed by running *Dnet.exe*. In particular, when WinMine/MSBN is used to learn the structure of a discrete Bayesian network, it uses a score-based approach together with a greedy DAG-based search algorithm. The search for the optimal structure of the Bayesian networks starts with the model containing no edges; then edges are greedily added, deleted, and reversed until a local maximum is reached.

Table 1 summarizes the analyzed software according to the following characteristics:

- Constraint is associated with constraint-based learning algorithms.
- Score is associated with score-based learning algorithms, more precisely with score available for optimization.
- Missing is associated with missing treatment with specific reference to structure learning.
- GUI tells about graphical user interface availability.

Table 1 Summary of software to learn the structure of a Bayesian networks from data

Name	Constraint	Score	Missing	GUI	License	Op. System	Language	API	Updated
BANJO		BDe			F	OSX, W, L, S	Java		?
BayesiaLab		MDL	✓	✓	C	OSX, W, L	?		2019
BNC-Weka	ICS, CI	AIC, BDe, MDL, K2		✓	F	OSX, W, L	Java		?
Bayes server	PC		✓	✓	C	OSX, W, L	Java	✓	2019
B-course		✓		✓	F	web service			?
BKD/BD		✓	✓	✓	F/C	W	?		?
blip		BDeu, BIC, K2	✓		F	OSX, W, L	Java/R		2019
bnlearn	PC, GS	BDe, BDs, BIC, MDL, K2	✓		F	OSX, W, L	R		2018
BNJ		✓		✓	F	W, L	Java		2004
bnstruct		AIC, BDeu, BIC			F	OSX, W, L	R		2018
BNT	IC, FCI, PC	BIC			F	OSX, W, L	MATLAB		2014
DEAL	✓	Bayes factor			F	OSX, W, L	R		2018
ELVIRA	PC	K2	✓	✓	F	OSX, W, L	Java		2010
Free BN	PC	K2			F	W, L	Java	✓	2011
GeNie and smile	PC	BS, GTT		✓	F/C	OSX, W, S	?	✓	2019
GOBNILP		BDeu			F	OSX, W	C	✓	2018
Hugin expert	PC, NPC	AIC, BIC	✓	✓	F/C	OSX, W, L	Java	✓	2019
LibB		BDe, BIC	✓		F	W, L	?		?
Libpgm		✓			F	OSX, W, L	Python		2012
OpenMarkov	PC	K2		✓	F	W	Java	✓	2012
PNL and PyPNL		AIC, BIC			F	OSX, W, L	C++, Python		2017
Pomegranate		MDL			F	OSX, W, L	C++, Python	✓	2017
ProBT		AIC, BIC, MDL, MI			C	OSX, W, L	C++	✓	2019
Tetrad	PC, FCI	BIC, BDeu	✓	✓	F	OSX, W, L	Java		2017
UnBBayes		K2		✓	F	OSX, W, L	Java		2010
WinMine/MSBN		K2			F	W	?		2006

✓ for **Constraint** and **Score** means *yes* but not the specific algorithm is unknown, ✓ for **Missing**, **GUI** and **API** means *yes*, F/C in **License** stand for Free and Commercial, OSX, W, L and S in **Op. System** stand, respectively, for OSX, Windows, Linux and Solaris, while ? stands for unknown

– License is about whether the software is free (F) or commercial (C).
– Op. System is about which operating system/s is/are supported.
– Language is associated with the programming language used to develop the software.
– API tells about the API availability.
– Updated tells the date of the last known update.

5 Guidelines

From a practitioner's point of view, it can be difficult to choose a method to use in a particular problem given the wide variety of available algorithms for inducing the structure of a Bayesian network. First of all, the nature of the problem to solve must be taken into account. If we are facing a classification or regression problem, then one should stick to the alternatives described in Sect. 3.5, because those structures are especially optimized for the prediction tasks.

If instead we are interested in knowing the independence relations between the variables in the model, score-based and constraint-based methods are more appropriate, since they are more expressive in terms of representing independences. It is important to point out that the methods explored in this paper do not take causality into account. It means that links between the variables in the learned structures should not be interpreted as causal relationships. Learning causal models is a research line that is receiving important attention in the last years. The work by He et al. [30] constitutes a good review of existing methods for these scenarios.

From a technical point of view, the features of the available data also determine the kinds of methods that can be used. More precisely, depending on the existence of missing data or the presence of continuous variables, the procedures reviewed in Sects. 3.3 and 3.4 become relevant, as they are able to handle such situations directly. Otherwise, a preprocessing step aimed to input the missing data or discretize the continuous variables would be necessary.

The size of the dataset is also relevant because it can determine the accuracy of the constraint-based methods. It is known that the power of the statistical tests grows with the sample size (i.e., the probability of accepting the null hypothesis when it is false decreases as the sample size increases). Therefore, test-based algorithms such as the PC have a tendency to include too many false edges when the sample size is large. A solution to this problem was developed by Bacciu et al. [6].

Finally, the software availability issue is crucial when attempting to apply any of the reviewed methods to practical problems. In that sense, the analysis of software solutions in Sect. 4 can be of help.

6 Concluding remarks

In this paper, we have presented a review of the most relevant literature on Bayesian network structure learning. We have classified the analyzed methods according to the approach they follow for solving the problem and discussed relevant issues about how they handle missing values or continuous variables. Methods oriented to specific tasks as classification and regression have also been accounted for. A thorough revision of available software tools has also been carried out.

Regarding future lines of research, there is an increasing interest of developing scalable methods able to handle large volumes of data and to run in distributed computational environments. Causality is also receiving considerable attention. This fact is connected to the need of developing models that are interpretable by humans.

Acknowledgements This work has been partly supported by the Spanish Ministry of Science, Innovation and Universities, grant TIN2016-77902-C3-3-P and by ERDF (FEDER) funds.

References

1. Abellán, J., Gómez-Olmedo, M., Moral, S.: Some variations on the PC algorithm. In: Third European Workshop on Probabilistic Graphical Models, pp. 1–8 (2006)
2. Adel, T., de Campos, C.P.: Learning Bayesian networks with incomplete data by augmentation. In: *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, pp. 1684–1690 (2017)
3. Alonso-Barba, J., de la Ossa, L., Gámez, J., Puerta, J.: Scaling up the greedy equivalence search algorithm by constraining the search space of equivalence classes. Int. J. Approx. Reason. **54**, 429–451 (2013)
4. Alonso-Barba, J.I., de la Ossa, L., Puerta, J.M.: Structural learning of Bayesian networks using local algorithms based on the space of orderings. Soft Comput. **15**(10), 1881–1895 (2011)
5. Alonso, J., de la Ossa, L., Gámez, J., Puerta, J.: On the use of local search heuristics to improve GES-based Bayesian network learning. Appl. Soft Comput. **64**, 366–376 (2018)
6. Bacciu, D., Etchells, T., Lisboa, P., Whittaker, J.: Efficient identification of independence networks using mutual information. Comput. Stat. **28**, 621–646 (2013)
7. Ben-Daya, M., Al-Fawzan, M.: A tabu search approach for the flow shop scheduling problem. Eur. J. Oper. Res. **109**(1), 88–95 (1998)
8. Bøttcher, S.: Learning Bayesian networks with mixed variables. In: *Proceedings of the Eighth International Workshop in Artificial Intelligence and Statistics* (2001)
9. Bøttcher, S., Dethlefsen, C.: deal: A package for learning bayesian networks. J. Stat. Softw. **8**, 1–40 (2003)
10. Buntine, W.: Theory refinement on Bayesian networks. In: *Proceedings of the 8th Conference on Uncertainty in Artificial Intelligence*, pp. 52–60 (1991)
11. Cheng, J., Bell, D.A., Liu, W.: An algorithm for Bayesian belief network construction from data. In: Proceedings of Artificial Intelligence and Statistics, pp. 83–90 (1997)
12. Chickering, D.: A transformational characterization of equivalent Bayesian network structures. In: *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence*, pp. 87–98. Morgan Kaufmann (1995)

13. Chickering, D.M., Heckerman, D., Meek, C.: Large-sample learning of Bayesian networks is NP-Hard. J. Mach. Learn. Res. **5**, 1287–1330 (2014)
14. Colombo, D., Maathuis, M.H.: Order-independent constraint-based causal structure learning. Journal of Machine Learning Research **15**, 3741–3782 (2014)
15. Consortium, Elvira.: Elvira: An environment for creating and using probabilistic graphical models. In: Gámez, J., Salmerón, A. (eds) Proceedings of the First European Workshop on Probabilistic Graphical Models, pp. 222–230 (2002)
16. Cooper, G.F.: The computational complexity of probabilistic inference using Bayesian belief networks. Artif. Intell. **42**, 393–405 (1990)
17. Cooper, G.F., Herskovits, E.: A Bayesian method for the induction of probabilistic networks from data. Mach. Learn. **9**, 309–347 (1992)
18. Cussens, J.: Bayesian network learning with cutting planes. In: Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence, pp. 153–160 (2011)
19. Cussens, J., Malone, B., Yuan, C.: IJCAI 2013 tutorial on optimal algorithms for learning Bayesian networks (2013). https://sites.google.com/site/ijcai2013bns/slides. Accessed June 2018
20. de Campos, C.P., Corani, G., Scanagatta, M., Cuccu, M., Zaffalon, M.: Learning extended tree augmented naive structures. Int. J. Approx. Reason. **68**, 153–163 (2015)
21. de Campos, C.P., Ji, Q.: Efficient structure learning of Bayesian networks using constraints. J. Mach. Learn. Res. **12**, 663–689 (2011)
22. de Campos, C.P., Zeng, Z., Ji, Q.: Structure learning of Bayesian networks using constraints. In: Proceedings of the 26th International Conference on Machine Learning, pp. 113–120 (2009)
23. Elidan, G., Gould, S.: Learning bounded treewidth Bayesian networks. J. Mach. Learn. Res. **9**, 2699–2731 (2008)
24. Fernández, A., Nielsen, J.D., Salmerón, A.: Learning Bayesian networks for regression from incomplete databases. Int. J. Uncertain. Fuzziness Knowl. Based Syst **18**(1), 69–86 (2010)
25. Fernández, A., Pérez-Bernabé, I., Salmerón, A.: On Using the PC Algorithm for Learning Continuous Bayesian Networks: An Experimental Analysis, CAEPIA'13. Lecture Notes in Computer Science **8109**, 342–351 (2013)
26. Fernández, A., Salmerón, A.: Extension of Bayesian network classifiers to regression problems. In: Geffner, H., Prada, R., Alexandre, I.M., David, N. (eds) Advances in Artificial Intelligence—IBERAMIA 2008, Vol. 5290 of Lecture Notes in Artificial Intelligence, pp. 83–92. Springer (2008)
27. Friedman, N.: The Bayesian structural EM algorithm. In: *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pp. 129–138 (1998)
28. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian network classifiers. Mach. Learn. **29**, 131–163 (1997)
29. Hand, D.J., Yu, K.: Idiot's Bayes–not so stupid after all? Int. Stat. Rev. **69**(3), 385–398 (2001)
30. He, Y., Jia, J., Geng, Z.: Structural learning of causal networks. Behaviormetrika **44**, 287–305 (2017)
31. Heckerman, D., Geiger, D., Chickering, D.: Learning Bayesian networks: the combination of knowledge and statistical data. Mach. Learn. **20**, 197–243 (1995)
32. Jaakkola, T., Sontag, D., Globerson, A., Meila, M.: Learning Bayesian network structure using LP relaxations. In: Proceedings of the 13th International Conference on Artificial Intelligence and Statistics, pp. 358–365 (2010)
33. Jaeger, M.: Probabilistic decision graphs—combining verification and ai techniques for probabilistic inference. Int. J. Uncertain. Fuzziness Knowl. Based Syst. **12**, 19–42 (2004)
34. Kalisch, M., Bühlmann, P.: Estimating high-dimensional directed acyclic graphs with the PC-algorithm. J. Mach. Learn. Res. **8**, 613–636 (2007)
35. Koivisto, M.: Parent assignment is hard for the MDL, AIC, and NML costs. In: Proceedings of the 29th Annual Conference On Learning Theory, vol. 4005, pp. 289–303 (2016)
36. Koivisto, M., Sood, K.: Exact Bayesian structure discovery in Bayesian networks. J. Mach. Learn. Res. **5**, 549–573 (2004)
37. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT Press, Boston (2009)
38. Korhonen, J., Parviainen, P.: Exact learning of bounded treewidth Bayesian networks. In: Artificial Intelligence and Statistics, pp 370–378 (2013)
39. Kwisthout, J. H.P., Bodlaender, H.L., van der Gaag, L.C.: The necessity of bounded treewidth for efficient inference in Bayesian networks. In: Proceedings of the 19th European Conference on Artificial Intelligence, pp. 237–242 (2010)
40. Lauritzen, S., Wermuth, N.: Graphical models for associations between variables, some of which are qualitative and some quantitative. Ann. Stat. **17**, 31–57 (1989)
41. Lee, C., van Beek, P.: Metaheuristics for score-and-search Bayesian network structure learning. In: Proceedings of the 30th Canadian Conference on Artificial Intelligence, pp. 129–141 (2017)
42. Madsen, A.L., Jensen, F., Salmerón, A., Langseth, H., Nielsen, T.D.: A parallel algorithm for Bayesian network structure learning from large data sets. Knowl. Based Syst. **117**, 46–55 (2017)
43. Malone, B., Kangas, K., Järvisalo, M., Koivisto, M., Myllymäki, P.: Empirical hardness of finding optimal Bayesian network structures: algorithm selection and runtime prediction. Mach. Learn. **107**, 1–37 (2018)
44. Malone, B.M.: Learning optimal Bayesian networks with heuristic search. Ph.D. thesis, Mississippi State University (2012)
45. Moral, S., Rumí, R., Salmerón, A.: Mixtures of Truncated Exponentials in Hybrid Bayesian Networks. In: Benferhat, S., Besnard , P. (eds) Symbolic and Quantitative Approaches to Reasoning with Uncertainty, Vol. 2143 of Lecture Notes in Artificial Intelligence, pp. 156–167. Springer (2001)
46. Nie, S., de Campos, C.P., Ji, Q.: Learning bounded treewidth Bayesian networks via sampling. In: Proceedings of the 13th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, pp. 387–396 (2015)
47. Nie, S., Mauá, D.D., de Campos, C.P., Ji, Q.: Advances in learning Bayesian networks of bounded treewidth. Adv. Neural Inf. Process. Syst. **27**, 2285–2293 (2014)
48. Nielsen, J.D., Rumí, R., Salmerón, A.: Structural-EM for learning PDG models from incomplete data. Int. J. Approx. Reason. **51**(5), 515–530 (2010)
49. Parviainen, P., Farahani, H.S., Lagergren, J.: Learning bounded treewidth Bayesian networks using integer linear programming. In: Proceedings of the 17th International Conference on Artificial Intelligence and Statistics, pp. 751–759 (2014)
50. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Elsevier, Amsterdam (1988)
51. Pearl, J.: Causality: models, reasoning and inference. Econom. Theory **19**(46), 675–685 (2003)
52. Pearl, J., Verma, T.S.: A theory of inferred causation. Stud. Logic Found. Math. **134**, 789–811 (1995)
53. Pourret, O., Naïm, P., Marcot, B.: Bayesian Networks: A Practical Guide to Applications. Wiley, Hoboken (2008)
54. Robinson, R.W.: Counting Labeled Acyclic Digraphs, New Directions in the Theory of Graphs, pp. 28–43. Academic Press, New York (1973)
55. Romero, V., Rumí, R., Salmerón, A.: Learning hybrid Bayesian networks using mixtures of truncated exponentials. Int. J. Approx. Reason. **42**, 54–68 (2006)
56. Scanagatta, M., Corani, G., de Campos, C.P., Zaffalon, M.: Learning treewidth-bounded Bayesian networks with thousands of variables. Adv. Neural Inf. Process. Syst. **29**, 1462–1470 (2016)

57. Scanagatta, M., Corani, G., de Campos, C.P., Zaffalon, M.: Approximate structure learning for large Bayesian networks. Mach. Learn. **107**, 1–19 (2018)
58. Scanagatta, M., Corani, G., Zaffalon, M.: Improved local search in Bayesian networks structure learning. In:Proceedings of the 3rd International Workshop on Advanced Methodologies for Bayesian Networks, pp. 45–56 (2017)
59. Scanagatta, M., Corani, G., Zaffalon, M., Yoo, J., Kang, U.: Efficient learning of bounded-treewidth Bayesian networks from complete and incomplete data sets. Int. J. Approx. Reason. **95**, 152–166 (2018)
60. Scanagatta, M., de Campos, C.P., Corani, G., Zaffalon, M.: Learning Bayesian networks with thousands of variables. Adv. Neural Inf. Process. Syst. **28**, 1855–1863 (2015)
61. Schwarz, G.: Estimating the dimension of a model. Ann. Stat. **6**, 461–464 (1978)
62. Scutari, M.: Bayesian network constraint-based structure learning algorithms: Parallel and optimised implementations in the bnlearn R package. CoRR (2014). arXiv:1406.7648
63. Silander, T., Myllymaki, P.: A simple approach for finding the globally optimal Bayesian network structure. In: Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence, pp. 445–452 (2006)
64. Spirtes, P., Glymour, C.N., Scheines, R.: Causation, Prediction, and Search. MIT Press, Boston (2000)
65. Steck, H., Tresp, V.: Bayesian belief networks for data mining. University of Magdeburg, pp 145–154 (1996)
66. Teyssier, M., Koller, D.: Ordering-based search: a simple and effective algorithm for learning Bayesian networks. In: Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence, pp. 584–590 (2005)
67. Yuan, C., Malone, B.: An improved admissible heuristic for learning optimal Bayesian networks. In: Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence, pp. 924–933 (2012)
68. Yuan, C., Malone, B., Wu, X.: Learning optimal Bayesian networks using A* search. In: *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pp. 2186–2191 (2011)
69. Zheng, X., Aragam, B., Ravikumar, P., Xing, E.: DAGs with no tears: Continuous optimization for structure learning. In: Advances in Neural Information Processing Systems, pp. 9492–9503 (2018)