Pattern Recognition 91 (2019) 190–199

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Tractable learning of Bayesian networks from partially observed data

Marco Benjumeda[1,*], Sergio Luengo-Sanchez[1,*], Pedro Larrañaga, Concha Bielza

Computational Intelligence Group, Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid, Spain

ARTICLE INFO

Article history:
Received 24 May 2018
Revised 21 December 2018
Accepted 22 February 2019
Available online 23 February 2019

Keywords:
Structural expectation-maximization
Bayesian network
Incomplete data
Inference complexity
Structure learning

ABSTRACT

The majority of real-world problems require addressing incomplete data. The use of the structural expectation-maximization algorithm is the most common approach toward learning Bayesian networks from incomplete datasets. However, its main limitation is its demanding computational cost, caused mainly by the need to make an inference at each iteration of the algorithm. In this paper, we propose a new method with the purpose of guaranteeing the efficiency of the learning process while improving the performance of the structural expectation-maximization algorithm. We address the first objective by applying an upper bound to the treewidth of the models to limit the complexity of the inference. To achieve this, we use an efficient heuristic to search the space of the elimination orders. For the second objective, we study the advantages of directly computing the score with respect to the observed data rather than an expectation of the score, and provide a strategy to efficiently perform these computations in the proposed method. We perform exhaustive experiments on synthetic and real-world datasets of varied dimensionalities, including datasets with thousands of variables and hundreds of thousands of instances. The experimental results support our claims empirically.

1. Introduction

Bayesian networks (BNs) [1,2] are probabilistic graphical models that provide a compact and self-explanatory representation of multidimensional probability distributions. BNs have been successfully applied in several machine learning problems including supervised classification [3–6] and clustering [7,8]. A BN includes two components. The first component is the structure, a directed acyclic graph that encodes conditional independences among triplets of variables in the network. The second component is the set of parameters, i.e., the conditional probability distributions of each variable given its parents in the graph.

One of the main advantages of BNs is that, as generative models, they can answer all conditional probability queries involving the variables of the network. In supervised classification, if the value of a certain set of feature variables is missing, a BN can exactly obtain the a posteriori most likely class label given the observed predictors. However, learning BNs from incomplete data continues to be a challenging problem. Expectation-maximization (EM) [9–11] is the most widely used algorithm for learning a model in the presence of missing values. It is an iterative method comprised of two steps. First, the expectation step (E-step) completes the data using inference. Then, the maximisation step (M-step) estimates the maximum likelihood parameters of the model from the completed data. The EM algorithm repeats both steps until convergence. Friedman's structural EM (SEM) [12] extends the EM algorithm to simultaneously learn the structure and parameters of a BN from incomplete data. This method has been successfully applied in semi-supervised classification [13,14] and clustering [15,16] problems. One of its most celebrated features is that, by optimising an expectation of the score, the algorithm guarantees convergence to a local optimum of the objective function with respect to the observed data. Note that under the missing at random assumption, optimising the scoring function with respect to the observed data is equivalent to optimising this function in the incomplete dataset. Because of its iterative nature, SEM is known to be computationally a highly demanding algorithm. Moreover, as inference in BNs is NP-hard [17], the computational cost of the E-step is likely to be prohibitive when the network candidates exhibit high inference complexity. Thus, bounding the inference complexity is critical to ensure the tractability of SEM.

The literature contains several approaches that address the problem of learning BNs with low inference complexity [18–20]. Nevertheless, these methods are not capable of learning from incomplete datasets. Most recently, Scanagatta et al. [21] proposed the k-MAX algorithm, a method for learning BNs with bounded treewidth. To learn from incomplete datasets, they introduced the SEM-kMAX algorithm in the M-step of SEM. To complete the

* Corresponding author.
E-mail addresses: marco.benjumeda.barquita@upm.es (M. Benjumeda), sluengo@fi.upm.es (S. Luengo-Sanchez), pedro.larranaga@fi.upm.es (P. Larrañaga), mcbielza@fi.upm.es (C. Bielza).

[1] Both authors contributed equally to this work.

https://doi.org/10.1016/j.patcog.2019.02.025
0031-3203/

data in each iteration (E-step), they use hard assignments because maintaining soft completions of the data in memory is unfeasible for their proposal. The main drawback of this approach is that unlike Friedman's SEM, it does not provide convergence guarantees on the objective function with respect to the observed data. Section 2.2 discusses the differences between hard and soft EM in greater detail.

In this paper, we propose an efficient method (i.e., with polynomial cost in the number of variables and number of instances of the dataset) for learning BNs with low inference complexity in the presence of missing values. For this purpose, we provide a rapid heuristic to estimate the upper bounds in the complexity of the models. Furthermore, we develop an efficient strategy to directly optimise the score with respect to the observed data and discuss its benefits compared to optimising an expectation of the score (e.g., Friedman's SEM). In the experiments, the proposed approach demonstrates promising results in terms of model fitting and imputation accuracy.

The remainder of this paper is organised as follows: Section 2 reviews previous work on inference complexity and learning BNs with incomplete data. Section 3 presents our proposal and provides theoretical results on the complexity of the algorithm. Section 4 provides the experimental results, comparing our proposal with Friedman's SEM and SEM-kMAX. Section 5 draws conclusions and recommends future research lines.

The software of the proposed method is freely available at https://github.com/sergioluengosanchez/TSEM.

2. Background

2.1. Inference complexity in Bayesian networks

A BN $\mathcal{B}$ represents a joint probability distribution over a set of discrete random variables $\mathcal{X} = \{X_1, \ldots, X_n\}$. A BN is a pair $\mathcal{B} = (\mathcal{G}, \boldsymbol{\theta})$, where $\mathcal{G}$ is the structure and $\boldsymbol{\theta}$ is the set of parameters $\Pr(X_i|\mathbf{Pa}_{X_i}^{\mathcal{G}})$ that represent the conditional probability distributions (CPDs) of each $X_i \in \mathcal{X}$. We use $\mathbf{Pa}_{X_i}^{\mathcal{G}}$ to denote the parents of node X_i in $\mathcal{G}$. Structure $\mathcal{G}$ encodes conditional independences among triplets of variables (local Markov property), i.e., each variable X_i is independent of its non-descendants given its parents $\mathbf{Pa}_{X_i}^{\mathcal{G}}$. Hence, the joint probability distribution can be factorised as

$$\Pr(X_1, \ldots, X_n) = \prod_{i=1}^{n} \Pr(X_i|\mathbf{Pa}_{X_i}^{\mathcal{G}}).$$

BNs can address multiple probabilistic inference problems such as evidence propagation, determination of the maximum a posteriori (MAP) hypothesis, and computation of the most probable explanation (MPE).

Evidence propagation includes the determination of the posterior probability of a set of query variables depending on specified evidence. This problem is typically NP-hard in BNs [17]. The majority of exact methods are based on variable elimination (VE) [22,23], recursive conditioning [24,25], and junction tree belief propagation [26,27].

VE is one of the most straightforward methods for inference in BNs. It successively eliminates the variables of a network until it yields the answer to a specified query. This algorithm is typically defined in terms of factors. A factor is a function that maps value assignments of a set of random variables to real positive numbers; CPDs are an example of factors. The elimination of a variable X_i includes outputting the product of all the factors containing X_i and marginalising out X_i. The order in which the variables are removed is called the elimination order (EO). Inference complexity is influenced by the selection of the EO [28]. To provide a formal definition of optimal EO, we must first introduce the concept of cluster [29]:

Definition 1. Let $\phi_1, \ldots, \phi_n$ be the sequence of factors induced by an EO π in graph $\mathcal{G}$. Cluster C_i is defined as the set of random variables in the domain of factor ϕ_i.

The optimality of an EO depends on its width.

Definition 2. Let $\mathbf{C} = (C_1, \ldots, C_n)$ be the sequence of clusters induced by an EO π in graph $\mathcal{G}$. The width of π in $\mathcal{G}$, which we denote as $\text{width}(\mathcal{G}, \pi)$, is the size of its largest cluster in $\mathbf{C}$ minus one. We refer to the EO with the minimal width for $\mathcal{G}$ as the optimal EO.

Note that our use of optimal EO is analogous to optimal graph triangulations [29], and we do not consider the best EO for specific inference queries. Inference complexity in BNs is typically evaluated in terms of their treewidth.

Definition 3. The treewidth of a graph $\mathcal{G}$ is the width of the optimal EO for $\mathcal{G}$.

The notion of treewidth was introduced by Robertson and Seymour [30]. Intuitively, the treewidth of a BN $\mathcal{B}$ can be understood as a measure of similarity between $\mathcal{B}$ and a tree (e.g., a tree has treewidth one). The computational cost of VE, recursive conditioning, and junction trees is exponential in the treewidth of $\mathcal{B}$. Thus, bounding the treewidth of $\mathcal{B}$ entails limiting its inference complexity [31].

Determining the MAP consists of a search for the most probable configuration of a set of variables in a BN for a specified evidence. The MPE is a special case of the MAP that involves a search for the most probable configuration of all non-instantiated variables. Both MAP and MPE are typically NP-hard problems [32]. Nevertheless, MPE can be computed in polynomial time if the treewidth of $\mathcal{B}$ is bounded [33], whereas MAP can be intractable even in models with bounded treewidth [34].

Computing the treewidth of a BN is NP-hard [35] and exact methods are exponential in the number of variables [36,37]. As mentioned above, the treewidth of a BN is specified by the width of its optimal EO. Several well-known heuristics exploit this relation to estimate the treewidth by searching for effective EOs. A collection of the most well-known approaches was overviewed by Bodlaender and Koster [38]; these include greedy search [39,40], chordal graph recognition heuristics [41,42], local search [43], and evolutionary computation techniques [44].

2.2. Learning Bayesian networks with partially observed data

Learning the BN structure is typically performed by a scoring metric that evaluates each candidate network with the data. When data is complete, the decomposability property of BN scoring functions such as Akaike information criterion (AIC) [45], Bayesian information criterion (BIC) [46], Bayesian Dirichlet for likelihood-equivalence (BDe) [47], and K2 [48] allows for efficient learning algorithms based on local search methods [48–51]. Evaluating a structure according to any of the above scores involves estimating the optimal parameters for each network candidate, which can be achieved efficiently when the data is complete. However, in the presence of missing values or hidden variables, it is not feasible to efficiently estimate the parameters because the network score does not decompose.

The most popular optimisation method for estimating the parameters from partially observed data is the EM algorithm [9–11]. EM addresses the missing data problem by selecting a starting point, which is either an initial set of parameters or an initial assignment to the missing variables. Once we have a parameter set,

we can apply inference to complete the data, or conversely, once we have the complete data, we can estimate the set of parameters using maximum likelihood estimation (MLE). EM iterates between both steps until convergence.

Assume we have a BN $\mathcal{B} = (\mathcal{G}, \boldsymbol{\theta})$ over a set of variables $\mathcal{X}$, and let a dataset $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$. Let $\mathbf{O}[m]$ be the variables whose values are observed (not missing) at the m-th instance of $\mathcal{D}$. EM uses the set of parameters $\boldsymbol{\theta}$ to complete the data probabilistically (E-step), resulting in the dataset $\mathcal{D}^+$. Obtaining $\mathcal{D}^+$ can be unfeasible in the majority of cases because its cost is exponential to the number of missing values. Nevertheless, the expected sufficient statistics (ESS) required for estimating a new set of parameters are obtained as follows:

$$\mathrm{ESS}_{\boldsymbol{\theta}}[X = x, \mathbf{Pa}_X^{\mathcal{G}} = \boldsymbol{u}] = \sum_{m=1}^{M} \Pr(x, \boldsymbol{u} | \mathbf{o}[m], \boldsymbol{\theta}), \tag{1}$$

where $\mathbf{o}[m]$ is the assignment of $\mathbf{O}[m]$ in $\mathcal{D}$ and $\mathrm{ESS}_{\boldsymbol{\theta}}[X = x, \mathbf{Pa}_X^{\mathcal{G}} = \boldsymbol{u}]$ are the ESS, i.e., the expected counts in $\mathcal{D}$ of variable X and $\mathbf{Pa}_X^{\mathcal{G}}$ according to $\boldsymbol{\theta}$. Eq. (1) requires the performance of inference for each of the M instances in the worst case when all the instances contain missing values. The new set of parameters is estimated from the completed dataset (M-step):

$$\hat{\boldsymbol{\theta}}_{x|\boldsymbol{u}} = \frac{\mathrm{ESS}_{\boldsymbol{\theta}}[x, \boldsymbol{u}]}{\mathrm{ESS}_{\boldsymbol{\theta}}[\boldsymbol{u}]}, \tag{2}$$

where $\hat{\boldsymbol{\theta}}_{x|\boldsymbol{u}}$ are the estimated parameters of the CPD $\Pr(X|\mathbf{Pa}_X^{\mathcal{G}})$. Both steps are repeated iteratively, and the likelihood of the parameters given $\mathcal{D}$ improves until convergence. EM has also been studied in the context of Bayesian learning [10,52]. However, in this paper we focus on the frequentist approach.

In BNs [53], the EM algorithm assumes a fixed structure $\mathcal{G}$ and optimises only the component $\boldsymbol{\theta}$ of the pair $\mathcal{B} = (\mathcal{G}, \boldsymbol{\theta})$. When $\mathcal{G}$ is unknown, it is not straightforward to apply EM. SEM [12] extends EM to learn both the structure and parameters.

SEM includes structural learning in the M-step. Any score+search structure learning method (e.g., [48,50,51]) can be used for this purpose; however, the scoring function to be maximised must be of the following form:

$$\mathrm{score}(\mathcal{D}, \mathcal{B}) = \mathrm{score}(\mathcal{D}, (\mathcal{G}, \boldsymbol{\theta})) = \ell(\boldsymbol{\theta}|\mathcal{D}) - \mathrm{pen}(\mathcal{B}, \mathcal{D}), \tag{3}$$

where $\ell(\boldsymbol{\theta}|\mathcal{D})$ is the log-likelihood of $\boldsymbol{\theta}$ given $\mathcal{D}$ and $\mathrm{pen}(\mathcal{B}, \mathcal{D})$ is a penalty function that can depend on $\mathcal{B}$ and $\mathcal{D}$. SEM starts with a specified initial structure $\mathcal{G}_0$ and a set of parameters $\boldsymbol{\theta}_0$. At each iteration, SEM selects the model and parameters with the highest expected score given the previous assessment. The use of the expected score is motivated by the next inequality [2]:

$$\mathrm{score}(\mathcal{D}, \mathcal{B}) - \mathrm{score}(\mathcal{D}, \mathcal{B}_0) \geq \mathrm{score}(\mathcal{D}^+, \mathcal{B}) - \mathrm{score}(\mathcal{D}^+, \mathcal{B}_0), \tag{4}$$

where $\mathcal{D}^+$ is the result of completing dataset $\mathcal{D}$ with BN $\mathcal{B}_0$. Intuitively, if $\mathcal{B}$ has a greater expected score than the model used to complete the data $\mathcal{B}_0$, then the score improvement with respect to the observed data is guaranteed. Hence, Eq. (4) ensures that the SEM algorithm converges to a local optimum.

To compute the expected score of each BN candidate, the ESS (Eq. (1)) are required. When the structure changes, a new set of ESS must be obtained, which requires the performance of inference in each case. This can be severely computationally demanding if the inference complexity of the models is not bounded. Hereafter, we refer to this method as soft SEM.

A less-demanding alternative is to complete the data according to its most probable assignment (E-step) and to estimate the MLE parameters from the completed dataset (M-step) [54]. This strategy does not require the computation of ESS for each change in the BN. Rather, the scores of all the BN candidates can be computed directly from the completed dataset. We refer to this method as hard SEM. In hard SEM, the complete dataset $\mathcal{D}^+$ is obtained by imputing the MPE of the missing values given the observed values $\mathbf{o}[m]$ for each instance m of $\mathcal{D}$:

$$\mathbf{h}[m] = \arg\max_{\mathbf{h}[m]} \Pr(\mathbf{h}[m] | \mathbf{o}[m], \boldsymbol{\theta}), \tag{5}$$

where $\mathbf{h}[m]$ is an assignment of the missing values at the m-th instance of $\mathcal{D}$.

The main difference between hard and soft SEM is that the former optimises over two objectives. In the E-step over the data completion $\mathcal{D}^+$ (see Eq. (5)) and in the M-step over the model $\max_{\mathcal{B}} \mathrm{score}(\mathcal{D}^+, \mathcal{B})$. Unlike soft SEM, the model selected by hard SEM after the M-step is not guaranteed to have a greater score with respect to the observed data than the previous candidate.

Limiting the inference complexity of the models is key to executing SEM efficiently. With this objective, Scanagatta et al. [21] adopt the k-MAX algorithm in the M-step of hard SEM. They call the resulting method SEM-kMAX.

k-MAX consists of two parts. First, a set of promising parents is identified using an approximation of the scoring function. Then, the structure incrementally grows according to a specified order of variables selected heuristically, ensuring that at each step, the treewidth bound is not exceeded. The theoretical computational cost of both parts is combinatorial in the number of variables and treewidth bound. In practice, they choose to set a predefined maximum execution time to explore the space of the solutions. Their experiments suggest that this strategy is more effective than other state-of-the-art methods when data is completely observed.

A relevant detail regarding the implementation of SEM-kMAX is that it adopts hard SEM rather than soft SEM. This is motivated by the unfeasibility of determining in advance what statistics will be required during the structure search. As k-MAX precomputes the scores beforehand at each iteration, in the majority of cases, it would be difficult to store the required number of sufficient statistics in memory. Scanagatta et al. [21] demonstrate that SEM-kMAX obtains promising results in the imputation experiments, yielding similar imputation accuracy to other well-known imputation methods in significantly less time.

3. Tractable SEM

This section introduces our proposal for learning BNs with low inference complexity from incomplete datasets. Algorithm 1 is

Algorithm 1: Pseudocode of Tractable SEM.

Input: Dataset $\mathcal{D}$, treewidth bound t_b
Output: Best BN structure $\mathcal{G}^*$, parameters $\boldsymbol{\theta}^*$ and EO π^*

select $\mathcal{G}_0$, $\boldsymbol{\theta}_0$, and π_0;
loop for $j = 0, 1, \ldots$ until convergence
  let $\mathcal{D}^+_{j+1}$ be the completed dataset given $\mathcal{D}$ and $\boldsymbol{\theta}_j$;
  let $\boldsymbol{\theta}_{j+1}$ be $\arg\max_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}|\mathcal{D}^+_{j+1})$;
  $\mathcal{G}_{j+1}, \pi_{j+1} \leftarrow \mathcal{G}_j, \pi_j$;
  let $c_1, \ldots, c_l$ be the local changes (i.e., arc additions, removals, or reversals) that canbe applied to $\mathcal{G}_j$;
  loop for $d = 1, \ldots, l$
    let $\mathcal{G}'$ be the result of applying c_d to $\mathcal{G}_j$;
    search for a low-width EO π' for $\mathcal{G}'$;
    if $\mathrm{width}(\mathcal{G}', \pi') \leq t_b$ **then**
      let $\boldsymbol{\theta}'$ be $\arg\max_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}|\mathcal{D}^+_{j+1})$;
      if $\mathrm{score}(\mathcal{D}_{j+1}, (\mathcal{G}', \boldsymbol{\theta}')) > \mathrm{score}(\mathcal{D}_{j+1}, (\mathcal{G}_{j+1}, \boldsymbol{\theta}_{j+1}))$ **then**
        $\mathcal{G}_{j+1}, \boldsymbol{\theta}_{j+1}, \pi_{j+1} \leftarrow \mathcal{G}', \boldsymbol{\theta}', \pi'$;
$\mathcal{G}^*, \boldsymbol{\theta}^*, \pi^* \leftarrow \mathcal{G}_{j+1}, \boldsymbol{\theta}_{j+1}, \pi_{j+1}$;

based on the SEM algorithm; however, it implements several changes toward guaranteeing efficient learning complexity and improving the model fitting with respect to the observed data. To ensure the tractability of Algorithm 1, it must limit the inference complexity of the BN candidates during the structure search. This is achieved by setting an upper bound t_b on the treewidth of the BN candidates. As the treewidth is the width of the optimal EO (see Section 2.1), we search for low-width EOs to obtain tight upper bounds in the treewidth of the models. Although determining optimal EOs is NP-hard, Section 3.1 presents an efficient heuristic for searching for low-width EOs that we later use as treewidth estimates. With the objective of improving the performance of soft SEM, Algorithm 1 computes the score directly with respect to the observed data rather than the expected score to compare BN candidates. In Section 3.2, we analyse in more detail the advantages and difficulties of this approach.

Algorithm 1 performs as follows. In line 1, the BN structure $\mathcal{G}_0$, parameters $\boldsymbol{\theta}_0$, and the EO π_0 are initialised. Line 2 is the main loop of the algorithm, which iterates until convergence. Lines 3 (E-step) and 4 (M-step) perform a single iteration of the EM algorithm with the purpose of updating the parameters of the current BN candidate. The E-step and M-step are computed according to Eqs. (1) and (2), respectively. In lines 6–13, Algorithm 1 searches for the local change that improves the score the most with respect to the observed data and that satisfies the treewidth bound t_b. For each BN structure candidate $\mathcal{G}'$, Algorithm 1 searches for a low-width EO (line 9). The width of the chosen EO is used as an estimate of the treewidth of $\mathcal{G}'$. In line 10, $\mathcal{G}'$ is rejected if this estimate is greater than t_b. Otherwise, the parameters of the new structure $\mathcal{G}'$ are updated with the completed dataset D^+_{j+1} (line 11) using MLE (Eq. (2)). Finally, the resultant BN $(\mathcal{G}', \boldsymbol{\theta}')$ is compared with the current best candidate (line 12), and the model with the greater score for $\mathcal{D}$ is selected (line 13).

Theorem 1. *If the method used for searching low-width EOs at line 9 of Algorithm 1 executes efficiently and the treewidth bound t_b is a constant (which is assumed to be small), then each iteration of Algorithm 1 consumes polynomial time in the number of variables and number of instances in dataset $\mathcal{D}$.*

Proof. Maintaining the probabilistic completion $\mathcal{D}^+$ of the data requires exponential time and space to the number of missing values for each data case. Alternatively, the ESS of $\mathcal{D}^+$ can be computed prior to the parameter estimations at lines 4 and 11. Computing the ESS of each BN candidate requires performing M inference queries, one per each data instance. Computing the score for $\mathcal{D}$ (line 12) also demands performing M inference queries. Note that, although inference is NP-hard in the general case, when the treewidth of the models is bounded by a constant, it can be performed efficiently. In one iteration of Algorithm 1, there are a maximum of $\mathcal{O}(n^2)$ network candidates (i.e., one for each possible local change). Therefore, the total number of inference queries required is upper-bounded by $\mathcal{O}(n^2 M)$.

The maximum likelihood parameters for a structure $\mathcal{G}$ and the completed dataset $\mathcal{D}^+$ can also be computed in polynomial time (lines 4 and 11). The number of possible local changes in a graph $\mathcal{G}$ is quadratic in the number of nodes. Thus, the loop at line 7 uses a number of iterations l that is also quadratic in n. The width of an EO (line 10) can be obtained in polynomial time [29]. Finally, if the process of searching for low-width EOs consumes polynomial time (line 9), each iteration can be computed efficiently. □

Theorem 1 ensures that each iteration of tractable SEM is computed efficiently under the described constraints. This implies that if Algorithm 1 loops for a polynomial number of iterations, the complete process is performed efficiently. Corollary 1 provides guarantees on the computational complexity of Algorithm 1.

Corollary 1. *If Algorithm 1 satisfies the conditions described in Theorem 1, the stopping condition for the loop at line 2 is $\mathcal{G}_{j+1} = \mathcal{G}_j$ and the local changes considered at line 5 are limited to arc additions, then, the complexity of Algorithm 1 is polynomial in the number of variables and the number of instances in the dataset.*

Proof. From Theorem 1, each iteration of Algorithm 1 must consume polynomial time. As the stopping criterion is $\mathcal{G}_{j+1} = \mathcal{G}_j$, the maximum number of iterations of the loop at line 2 is the number of local changes that can be applied to $\mathcal{G}$. Given that the local changes are limited to arc additions, and a complete graph of n nodes has $\frac{n(n-1)}{2}$ arcs, the maximum number of iterations that this algorithm could perform is upper-bounded by n^2. Thus, if the above conditions are fulfilled, Algorithm 1 performs a polynomial number of iterations, each in polynomial time. Hence, its complexity is polynomial in n and M. □

From Theorem 1 and Corollary 1, Algorithm 1 can only be executed in polynomial time if the method used for searching low-width EOs is also polynomial. Section 3.1 proposes an efficient method for this purpose.

3.1. Efficient search of low-width elimination orders

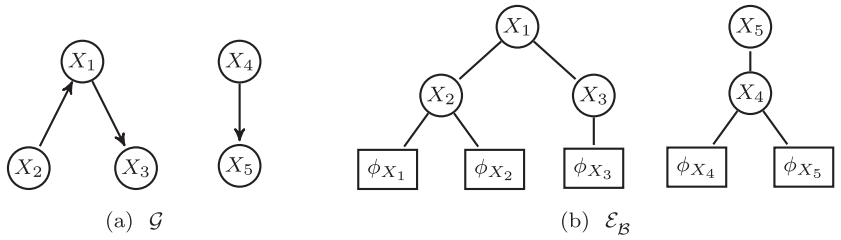
The time complexity of tractable SEM is highly sensitive to the computational cost of searching for low-width EOs. Benjumeda et al. [55] demonstrated that elimination trees (ETs) [56] can be used to implement an efficient heuristic for this problem. In this section, we apply their approach to search for low-width EOs.

In a BN $\mathcal{B}$ over a set of n random variables $\mathcal{X}$, there are $n!$ EOs of $\mathcal{X}$, although a number of them are typically equivalent for $\mathcal{B}$. Two EOs are equivalent for a BN $\mathcal{B}$ if they induce identical factors during VE. For example, assume a BN $\mathcal{B}$ over variables $\mathcal{X}$ that represents the product of the marginals $Pr(X_1, \ldots, X_n) = Pr(X_1)Pr(X_2)\cdots Pr(X_n)$. Given $\mathcal{B}$, VE induces identical factors for any EO of $X_1, \ldots, X_n$. Hence, all $n!$ possible EOs are equivalent for $\mathcal{B}$. Circumventing this redundancy reduces the size of the search space during the learning process. Grant and Horsch [56] proposed elimination trees (ETs), a representation for recursive conditioning, which has been adapted to exploit equivalence among EOs [55].

Definition 4. (Elimination tree) Let $\mathcal{B}$ be a BN over $\mathcal{X}$. An elimination tree $\mathcal{E}_\mathcal{B}$ over $\mathcal{X}$ is composed of an inner node (node with children) for each variable $X_i \in \mathcal{X}$ and a leaf node labelled ϕ_{X_i} that represents the parameters of $\mathcal{B}$ of each node $X_i \in \mathcal{X}$.

Note that an ET can be a forest when there are unconnected subgraphs in the BN. Each ET $\mathcal{E}_\mathcal{B}$ represents a set $\mathcal{S}$ of EOs such that in each EO, $\pi \in \mathcal{S}$, $(X_i < X_j)_\pi$ if $X_j \in \mathbf{Pred}_{X_i}^{\mathcal{E}_\mathcal{B}}$; here, $(X_i < X_j)_\pi$ denotes that X_i is eliminated before X_j in π and $\mathbf{Pred}_{X_i}^{\mathcal{E}_\mathcal{B}}$ is the set of predecessors of X_i in $\mathcal{E}_\mathcal{B}$. This precedence must be read from the root nodes (i.e., nodes without a parent) to the leaves. Fig. 1 displays an example of a set of EOs $\mathcal{S}$ represented by an ET $\mathcal{E}_\mathcal{B}$.

Algorithm 2 exploits the capability of ETs to prevent redundancy in EOs. A local change in the structure of $\mathcal{B}$ changes the conditions where two EOs are equivalent. Herein, we use Algorithm 2 incrementally. From the EO chosen in the previous iteration, it searches for EOs of lower width for $\mathcal{B}'$ (i.e., $\mathcal{B}$ after a local change). Intuitively, we believe that if a specified EO π is near to optimal for $\mathcal{B}$, it should not be far from optimal for $\mathcal{B}'$. Nevertheless, as further local changes are introduced, the quality of π is likely to degrade. Moreover, there are situations where a single local change can produce a significant change in the width of the EO. The ET optimisation heuristic proposed in [55] is used to mitigate this problem. The method (optimise_ET in Algorithm 2) receives an ET $\mathcal{E}_\mathcal{B}$ as input and greedily searches for lower-width ETs until no improvement is found.



(a) $\mathcal{G}$ (b) $\mathcal{E}_{\mathcal{B}}$

Fig. 1. (a) $\mathcal{G}$ is graph structure of BN $\mathcal{B}$; (b) $\mathcal{E}_{\mathcal{B}}$ is ET representing set of EOs $\mathcal{S}$ for $\mathcal{B}$. Given that in $\mathcal{E}_{\mathcal{B}}$, X_1 is a predecessor of X_2 and X_3 and that X_5 is a predecessor of X_4 in $\mathcal{E}_{\mathcal{B}}$, each $\pi \in \mathcal{S}$ fulfils the following: $(X_2 \prec X_1)_\pi$, $(X_3 \prec X_1)_\pi$, and $(X_4 \prec X_5)_\pi$. For example, $(X_2, X_3, X_1, X_4, X_5)$ and $(X_4, X_2, X_3, X_5, X_1)$ belong to $\mathcal{S}$. Note that $\mathcal{E}_{\mathcal{B}}$ is a forest rather than a tree because $\mathcal{G}$ is composed of two unconnected subgraphs.

Algorithm 2: Incremental search for a low-width EO.

Input: BN $\mathcal{B} = (\mathcal{G}, \boldsymbol{\theta})$, EO π, local change c
Output: EO π'

1 let $\mathcal{E}_{\mathcal{B}}$ be an unconnected ET;
2 let $C_1, \ldots, C_n$ be the sequence of clusters induced by π in $\mathcal{G}$;
3 **loop** for $i = 1, \ldots, n$
4 let $X_{\pi(i)}$ be the variable that corresponds to the i-th position in π;
5 **loop** for $k = 1, \ldots, n$
6 **if** $X_{\pi(i)} \in C_k$ and $\mathbf{Pred}_{X_{\pi(k)}}^{\mathcal{E}_{\mathcal{B}}} = \emptyset$ **then**
7 set $X_{\pi(i)}$ as the parent of $X_{\pi(k)}$ in $\mathcal{E}_{\mathcal{B}}$;
8 **if** $X_{\pi(i)}$ is in the domain of $\phi_{X_{\pi(k)}}$ and $\mathbf{Pred}_{\phi_{X_{\pi(k)}}}^{\mathcal{E}_{\mathcal{B}}} = \emptyset$ **then**
9 set $X_{\pi(i)}$ as the parent of $\phi_{X_{\pi(k)}}$ in $\mathcal{E}_{\mathcal{B}}$;
10 $\mathcal{E}'_{\mathcal{B}} \leftarrow$ optimise_ET($\mathcal{E}_{\mathcal{B}}$);
11 Let π' be an EO represented by $\mathcal{E}'_{\mathcal{B}}$;
12 **return** π'

First, Algorithm 2 obtains an ET $\mathcal{E}_{\mathcal{B}}$ that represents all the EOs equivalent to π for $\mathcal{B}$ (lines 1–9). Then, the variables are visited in the sequence given by π (line 3). When variable $X_{\pi(i)}$ is visited, node $X_{\pi(i)}$ is set as the parent of the nodes whose cluster C_k contains $X_{\pi(i)}$ and with an empty set of predecessors in $\mathcal{E}_{\mathcal{B}}$ (lines 3–9). The optimise_ET method (line 10) is a greedy heuristic that visits each node from the shallowest to the deepest, checking at each step whether exchanging the position of the visited node and its parent reduces the width of the ET. Finally, Algorithm 2 selects an EO (line 11) from the set of EOs $\mathcal{S}$ represented by $\mathcal{E}'_{\mathcal{B}}$.

As mentioned above, the time complexity of Algorithm 2 is the most important factor to ensuring the efficiency of Algorithm 1. Theorem 1 requires a polynomial method for determining EOs such that each iteration of tractable SEM consumes polynomial time. Proposition 1 demonstrates that Algorithm 2 fulfils this condition.

Proposition 1. *Algorithm 2 executes in polynomial time in the number of variables.*

Proof. Line 1 consumes linear time. Obtaining the cluster sequence at line 3 requires polynomial time [29]. The loops at lines 3 and 5 iterate over n values. The computational time required by lines 6 and 8 is less than $\mathcal{O}(n)$, given that the cardinalities of the domains of C_k and $\phi_{X_{\pi(k)}}$ are smaller than or equal to n. Hence, lines 3–9 consume polynomial time. The complexity of method optimise_ET (line 10) is upper bounded by $\mathcal{O}(n^2 \cdot \text{width}(\mathcal{G}, \pi))$ (see Lemma 10 in [55] for the proof). Obtaining an EO from an ET (line 11) is equivalent to retrieving a reverse topological ordering of the ET, which can be performed in polynomial time. □

3.2. Optimising the score with respect to the observed data

One of the most celebrated properties of soft SEM is that it ensures that the scoring function does not decrease with respect to the observed data $\mathcal{D}$ after each iteration. According to Eq. (4), given a model $\mathcal{B}_0$ and a dataset $\mathcal{D}^+$, where $\mathcal{D}^+$ is the probabilistic completion of $\mathcal{D}$ by $\mathcal{B}_0$, any improvement in the expected score ($score(\mathcal{D}^+, \mathcal{B}) \geq score(\mathcal{D}^+, \mathcal{B}_0)$) results in an improvement in the score with respect to the observed data ($score(\mathcal{D}, \mathcal{B}) \geq score(\mathcal{D}, \mathcal{B}_0)$). Thus, soft SEM ensures the convergence of the score with respect to the observed data to a local optimum. The advantage of using the expected score rather than the score with respect to the observed data is that once the data is completed during the E-step, the expected score decomposes and therefore the local changes in the network locally affect the nodes, facilitating more efficient learning (e.g., by using cache).

Despite the desirable properties of soft SEM, Eq. (4) offers no guarantee that the model selected at the end of the search is near to the optimum. For example, if a greedy search is used, only the first local change guarantees an improvement in the score for the observed data. Therefore, if we aim to ensure improvements on the score with respect to the observed data after applying every local change, the data must be repeatedly completed, preventing soft SEM from exploiting the decomposition of the score. Moreover, a local change that improves the score with respect to the observed data may not improve the expected score. If many local changes are incorrectly rejected, the learning process could terminate early.

These problems can be overcome by directly using the score with respect to the observed data. The bottleneck of Algorithm 1 is the inference required for the computation of the ESS and score for each network candidate (line 7). Although the cost of this process is polynomial in the number of variables n and the number of instances of the dataset M when the treewidth of the BN candidates is small, the computational time required to answer all the inference queries can be high when n or M are large.

Next, we propose a heuristic with the objective of reducing the number of inference queries during the learning process while ensuring that each local change applied improves the score with respect to the observed data. We modify Algorithm 1 as follows:

- At line 6, the data is completed with hard assignments and the local changes $c_1, \ldots, c_l$ are ordered according to their score for the completed dataset.
- The loop at line 7 terminates when the first local change c_d that improves the score with respect to the observed data is identified (i.e., we follow a best-first strategy).

Ordering the local changes as suggested above requires imputing the data once. Testing all the local changes until an improvement is identified prevents Algorithm 1 from falling into local optima in the early stages of the search. In Section 4, we evaluate this strategy. In the experiments, the proposed approach

Table 1
Basic properties of BNs used to generate synthetic datasets.

Dataset	N. vars	N. arcs	N. parameters
SACHS	11	17	178
ALARM	37	46	509
BARLEY	48	84	114,005
CHILD	20	25	230
INSURANCE	27	52	984
MILDEW	35	46	540,150
WATER	32	66	10,083
HAILFINDER	56	66	2,656
HEPAR II	70	123	1,453
WIN95PTS	76	112	574
PATHFINDER	135	200	77,155

outperformed soft SEM and SEM-kMAX in terms of fitting the data and imputation accuracy.

4. Experimental results

In this section, we empirically evaluate the performance of tractable SEM in terms of data fitting, imputation accuracy, and computational complexity. First, in Section 4.1, we compare the proposed approach with soft SEM to highlight the advantages of directly computing the score with respect to the observed data and the reduction in the computational cost provided by the strategy proposed in Section 3.2. Then, in Section 4.2, we compare tractable SEM with SEM-kMAX for analysing the performance of the proposed approach in real world datasets of varied dimensionalities.

The experiments below include two versions of tractable SEM: TSEM implements Algorithm 1 and uses the heuristic described in Section 3.2 to accelerate the learning process. TSEM-poly also fulfils the conditions required by Corollary 1 to ensure polynomial computational complexity.

For all the methods, the score function to maximise is BIC. We analysed the significance of the differences found for each performance measure in all experiments using the Friedman test with $\alpha = 0.05$ and Holm's [57] and Shaffer's [58] post-hoc procedures. Both Holm's and Shaffer's procedures associate pairwise comparisons with a set of hypotheses and perform a step-down process with the corresponding set of ordered p-values to adjust the value of α [59].

Experiments were performed on a computer with an Intel Core i7-6700K CPU at 4.00 GHz with 16 GB main memory, running Ubuntu 16.04 LTS. TSEM, TSEM-poly, and soft SEM were written in Python 2.7.12, and integrate specific functions developed in C++11 (version 5.4.0). SEM-kMAX was downloaded from http://ipg.idsia.ch/software/blip and is written in Java.

4.1. Comparison with soft SEM

This first experimental study highlights the differences between using TSEM, TSEM-poly, and soft SEM. The goal is to compare the models output using the three methods in diverse scenarios according to a set of performance measures. To compare these methods, we generated synthetic data from 11 real-world BNs. These BNs were obtained from the bnlearn BN repository http://www.bnlearn.com/bnrepository/, and are cited therein. Table 1 lists the number of variables (N. vars), arcs (N. arcs), and parameters (N. parameters) of each BN. To include a wide variety of scenarios, we generated training and testing datasets of different sample sizes (500, 2000, and 5000) and different percentages of missing values (30%, 50%, and 70%) from the above networks. In TSEM and TSEM-poly, we set the treewidth bound t_b to "5", which in our experience provides an acceptable trade-off between efficiency and expressiveness. In Section 4.2, we evaluate the performance of TSEM and TSEM-poly with different treewidth bounds.

Table 2
Comparison of methods in all datasets with 500 instances. Best results are denoted in boldface.

	TSEM	TSEM-poly	soft SEM
Mean rank BIC	**1.42 ± 0.5**	2.36 ± 0.55	2.21 ± 0.99
Mean rank LL	**1.3 ± 0.47**	2.27 ± 0.52	2.42 ± 0.9
Mean rank acc	**1.15 ± 0.36**	2.09 ± 0.38	2.76 ± 0.66
Mean rank time	1.82 ± 0.39	**1.18 ± 0.39**	3 ± 0

Table 3
Comparison of methods in all datasets with 2,000 instances. Best results are denoted in boldface.

	TSEM	TSEM-poly	soft SEM
Mean rank BIC	**1.15 ± 0.36**	2.18 ± 0.39	2.67 ± 0.74
Mean rank LL	**1.33 ± 0.48**	2.06 ± 0.61	2.61 ± 0.79
Mean rank acc	**1.21 ± 0.42**	1.97 ± 0.47	2.82 ± 0.58
Mean rank time	1.91 ± 0.38	**1.15 ± 0.36**	2.94 ± 0.35

Table 4
Comparison of methods in all datasets with 5,000 instances. Best results are denoted in boldface.

	TSEM	TSEM-poly	soft SEM
Mean rank BIC	**1.06 ± 0.24**	2.06 ± 0.24	2.88 ± 0.48
Mean rank LL	**1.12 ± 0.33**	1.94 ± 0.35	2.94 ± 0.35
Mean rank acc	**1.12 ± 0.33**	1.94 ± 0.35	2.94 ± 0.35
Mean rank time	1.97 ± 0.39	**1.12 ± 0.33**	2.91 ± 0.38

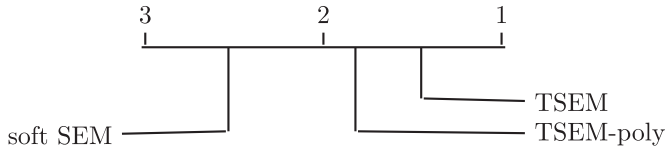


Fig. 2. Comparison of mean rank BIC scores in training dataset with Holm's and Shaffer's post-hoc procedures in synthetic datasets.

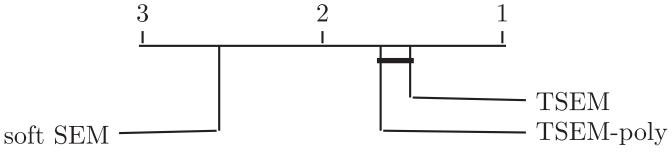


Fig. 3. Comparison of mean rank log-likelihood (LL) in test dataset with Holm's and Shaffer's post-hoc procedures in synthetic datasets.

Tables 2–4 display the experimental results of comparing the above approaches. We use the following performance measures: BIC is the BIC score of the models with respect to the observed values in the training dataset, LL is the log-likelihood of the models in the test dataset, acc is the imputation accuracy in the training dataset, and time is the learning time (in seconds). For each performance measure, the mean rank ± the standard deviation of each method over all the datasets is displayed. The ranking of the methods is given by their average performance (BIC, LL, acc, and time) compared to the others (i.e., the best is ranked first and the worst is ranked third). The detailed results are supplied as Supplementary Material.

Figs. 2–5 graphically present the results obtained with Holm's and Shaffer's procedures for each performance measure in all datasets. In the figures, groups of methods that are not significantly different are connected with a thick horizontal line. This is the graphical representation proposed by Demšar [60]. Each figure represents, in fact, both procedures, given that the significant

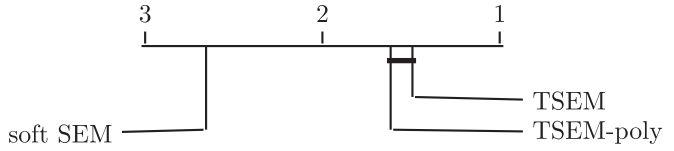


Fig. 4. Comparison of mean rank imputation accuracy (acc) in training dataset with Holm's and Shaffer's post-hoc procedures in synthetic datasets.

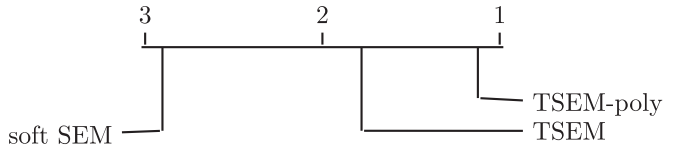


Fig. 5. Comparison of mean rank learning time in training dataset with Holm's and Shaffer's post-hoc procedures in synthetic datasets.

differences observed by Shaffer's procedure are identical to those observed by Holm's procedure.

In the case of BIC (see Fig. 2), significant differences were found among the methods. TSEM performed the best, followed by TSEM-poly and soft SEM. The results suggest that these differences are caused by the inability of the expected score to recognise many of the changes that improve BIC with respect to the observed data. For example, of all the local changes performed by TSEM, on average, only 19% improved the expected score. This leads TSEM to apply, on average, 33% more local changes than soft SEM. The rare situations where soft SEM achieved a greater BIC score than TSEM and TSEM-poly correspond to small datasets with a high percentage of missing values. In these situations, the BN structures output by all the methods were extremely sparse or even completely unconnected. As TSEM and TSEM-poly terminate when they do not find a local change that improves the BIC, both methods clearly require less time than soft SEM to optimise the parameters. A way of improving the performance of the proposed approach in these situations would be to perform the EM algorithm until convergence in the output models.

Figs. 3 and 4 compare the log-likelihood and imputation accuracy of the models, respectively. TSEM and TSEM-poly performed significantly better than soft SEM for both measures. However, no significant differences were found between TSEM and TSEM-poly.

Fig. 5 indicates significant differences among the learning times of all the methods; TSEM-poly was the fastest method overall, followed by TSEM and soft SEM. As these experiments were only performed in datasets generated from medium and small size BNs, the treewidth of the models learnt by soft SEM was never high (seven in the worst case). Therefore, the differences in the computational time can be explained by the number of times that each approach computed the parameters. For example, soft SEM required computing the ESS an average 61 more times than TSEM (details provided as Supplementary Material). Note that as soft SEM does not bound the treewidth of the models during the learning process, larger datasets typically lead to models with greater treewidth, where exact inference, and therefore computing the ESS is unfeasible.

4.2. Comparison with SEM-kmax

In this section, the proposed approach is compared with SEM-kMAX for learning bounded treewidth BNs from incomplete datasets. To perform the experiments, we used 22 real-world datasets of varied dimensionalities (N. vars) and sample sizes (N. train inst. and N. test inst.). These datasets were previously used in several papers [21,55,61–63][2]. Table 5 provides the number of variables and number of training and testing instances in each dataset.

We set several scenarios to evaluate both methods according to the performance measures used in Section 4.1. For each real-world dataset, we input, at random, different percentages of missing values (30%, 50%, and 70%), and tested the methods in all the situations. For each method, we learned a model using four different treewidth bounds (2, 3, 4, and 5). We set the parameters of SEM-kMAX to the values recommended by Scanagatta et al. [21]. Concretely, they set an execution time of n s (i.e., a second for each variable) to compute the cache of the most optimum parent sets and n/10 s for the structure search. Tables 6–9 display the

Table 5
Basic properties of real-world datasets.

Dataset	N. vars	N. train inst.	N. test inst.
NLTCS	16	16,181	3,236
MSNBC1	17	291,326	58,265
KDDCup	65	180,092	34,955
Plants	69	17,412	3,482
Audio	100	15,000	3,000
Jester	100	9,000	4,116
Netflix	100	15,000	3,000
Accidents	111	12,758	2,551
Retail	135	22,041	4,408
Pumsb-star	163	12,262	2,452
DNA	180	1,600	1,186
Kosarek	190	33,375	6,675
MSWeb	294	29,441	5,000
Book	500	8,700	1,739
EachMovie	500	4,525	591
WebKB	839	2,803	838
Reuters-521	889	6,532	1,540
20 NewsGroup	910	11,293	3,764
Movie reviews	1,001	1,600	250
BBC	1,058	1,670	330
Voting	1,359	1,214	350
Ad	1,556	2,461	491

Table 6
Comparison of methods in all datasets, using treewidth bound of "2". Optimal results are denoted in boldface.

	TSEM	TSEM-poly	SEM-kMAX
Mean rank BIC	**1.17 ± 0.38**	1.83 ± 0.38	3 ± 0
Mean rank LL	**1.35 ± 0.48**	1.65 ± 0.48	3 ± 0
Mean rank acc	**1.42 ± 0.58**	1.79 ± 0.54	2.79 ± 0.62
Mean rank time	1.94 ± 0.24	**1.06 ± 0.24**	3 ± 0

Table 7
Comparison of methods in all datasets, using treewidth bound of "3". Optimal results are denoted in boldface.

	TSEM	TSEM-poly	SEM-kMAX
Mean rank BIC	**1.17 ± 0.38**	1.83 ± 0.38	3 ± 0
Mean rank LL	**1.48 ± 0.5**	1.52 ± 0.5	3 ± 0
Mean rank acc	**1.55 ± 0.59**	1.68 ± 0.61	2.77 ± 0.63
Mean rank time	1.95 ± 0.21	**1.05 ± 0.21**	3 ± 0

Table 8
Comparison of methods in all datasets, using treewidth bound of "4". Optimal results are denoted in boldface.

	TSEM	TSEM-poly	SEM-kMAX
Mean rank BIC	**1.17 ± 0.38**	1.83 ± 0.38	3 ± 0
Mean rank LL	**1.35 ± 0.48**	1.65 ± 0.48	3 ± 0
Mean rank acc	1.67 ± 0.64	**1.64 ± 0.6**	2.7 ± 0.72
Mean rank time	1.85 ± 0.36	**1.15 ± 0.36**	3 ± 0

[2] The real-world datasets can be found at https://github.com/UCLA-StarAI/Density-Estimation-Datasets.

Table 9
Comparison of methods in all datasets, using treewidth bound of "5". Optimal results are denoted in boldface.

	TSEM	TSEM-poly	SEM-kMAX
Mean rank BIC	**1.21 ± 0.41**	1.79 ± 0.41	3 ± 0
Mean rank LL	**1.44 ± 0.5**	1.56 ± 0.5	3 ± 0
Mean rank acc	1.68 ± 0.68	**1.62 ± 0.55**	2.7 ± 0.72
Mean rank time	1.85 ± 0.36	**1.15 ± 0.36**	3 ± 0

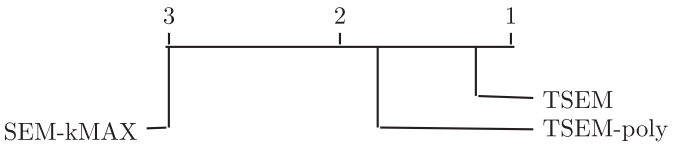


Fig. 6. Comparison of mean rank BIC scores in training dataset with Holm's and Shaffer's post-hoc procedures in real-world datasets.

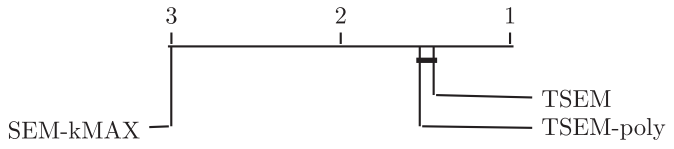


Fig. 7. Comparison of mean rank log-likelihood (LL) in test dataset with Holm's and Shaffer's post-hoc procedures in real-world datasets.

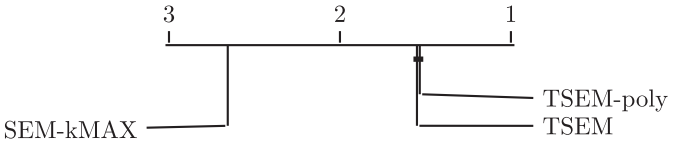


Fig. 8. Comparison of mean rank imputation accuracy (acc) in the training dataset with Holm's and Shaffer's post-hoc procedures in real-world datasets.

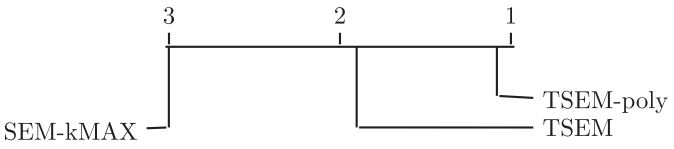


Fig. 9. Comparison of mean rank learning time in training dataset with Holm's and Shaffer's post-hoc procedures in real-world datasets.

mean rank ± the standard deviation (over all the datasets) of each method for every performance measure given a treewidth bound t_b. The detailed results are supplied as Supplementary Material.

The significance results obtained with Holm's and Shaffer's post-hoc procedures over all datasets and treewidth bounds are displayed in Figs. 6–9.

TSEM and TSEM-poly performed better than SEM-kMAX in all experiments in terms of BIC and LL, and achieved a significantly greater imputation accuracy. The treewidth of the models output by each method suggests that the proposed approach leads to a tighter treewidth bound fitting than SEM-kMAX (details are provided in the Supplementary Material). Although TSEM led to significantly greater BIC scores than TSEM-poly, the difference in the majority of cases was small. This explains why the results of both methods in terms of LL and imputation accuracy were similar. Finally, TSEM and TSEM-poly executed faster than SEM-kMAX in all experiments. However, we must be cautious when interpreting these results given that these methods were implemented in different programming languages. Although TSEM-poly executed faster than TSEM in the majority of cases, the differences in learning time in each particular case were small.

5. Conclusions and future research

In this study, we addressed the problem of learning BNs in the presence of missing values and hidden variables in tractable time. We proposed an adaptation of SEM that ensures the efficiency of the E-step by bounding the inference complexity of the BN candidates. To limit their inference complexity, we proposed the use of an efficiency-focused heuristic for searching for low-width EOs. We demonstrated that the resulting algorithm consumes polynomial time under certain conditions. Further, we analysed the advantages of using the score with respect to the observed data directly, rather than the expected score, and provided a heuristic to reduce the number of inference queries performed during the learning process.

As demonstrated by the experimental results, the proposed approach outperformed soft SEM and SEM-kMAX based on all the tested evaluation metrics. Apparently, these differences were a consequence of the advantages of directly optimising the score with respect to the observed data. Moreover, the proposed heuristics lead to a significant reduction in the computational cost of the learning process.

Friedman adapted soft SEM to the Bayesian learning of BNs [64], which entails several advantages. For example, it provides a method to incorporate prior knowledge and a superior evaluation of the generalization properties of a model given the data. Adapting our proposal to use Bayesian scoring functions would be relatively straightforward, and we intend to do this in the future.

At present, there is an increasing interest in learning with hidden variables from high dimensional spaces. Examples are multidimensional clustering [65,66] and learning deep probabilistic graphical models [67,68]. We consider that our proposal is effective for these tasks, and we intend to study its application to these problems.

Acknowledgements

This work was partially supported by the Spanish Ministry of Science and Innovation through the Cajal Blue Brain (C080020-09; the Spanish partner of the Blue Brain initiative from EPFL) and TIN2016-79684-P projects; and by the Regional Government of Madrid through the S2013/ICE-2845-CASI-CAM-CM project. This project received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 785907 (HBP SGA2). M. Benjumeda is supported by a predoctoral contract for the formation of doctors from the Spanish Ministry of Science and Innovation (BES-2014-068637).

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.patcog.2019.02.025.

References

[1] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann, San Francisco, CA, USA, 1988.
[2] D. Koller, N. Friedman, Probabilistic Graphical Models: Principles and Techniques, The MIT Press, Cambridge, MA, USA, 2009.
[3] C. Bielza, P. Larrañaga, Discrete Bayesian network classifiers: a survey, ACM Comput. Surv. 47 (1) (2014) 5.
[4] L. Liu, S. Wang, G. Su, Z.-G. Huang, M. Liu, Towards complex activity recognition using a Bayesian network-based probabilistic generative framework, Pattern Recognit. 68 (2017) 295–309.
[5] P. Peng, Y. Tian, Y. Wang, J. Li, T. Huang, Robust multiple cameras pedestrian detection with multi-view Bayesian network, Pattern Recognit. 48 (5) (2015) 1760–1772.

[6] Y. Li, S.M. Mavadati, M.H. Mahoor, Y. Zhao, Q. Ji, Measuring the intensity of spontaneous facial action units with dynamic Bayesian network, Pattern Recognit 48 (11) (2015) 3417–3427.
[7] J.M. Peña, J.A. Lozano, P. Larrañaga, Learning Bayesian networks for clustering by means of constructive induction, Pattern Recognit. Lett. 20 (1999) 1219–1230.
[8] D.T. Pham, G.A. Ruz, Unsupervised training of Bayesian networks for data clustering, Proc. R. Soc. London A 465 (2109) (2009) 2927–2948.
[9] A. Dempster, N. Laird, D. Rubin, Maximum likelihood from incomplete data via the EM algorithm, J. R. Stat. Soc. Ser. B 1 (1977) 1–38.
[10] G. McLachlan, T. Krishnan, The EM algorithm and extensions, John Wiley & Sons, New York City, NY, USA, 2008.
[11] W. Liao, Q. Ji, Learning Bayesian network parameters under incomplete data with domain knowledge, Pattern Recognit. 42 (11) (2009) 3046–3056.
[12] N. Friedman, Learning belief networks in the presence of missing values and hidden variables, in: Proceedings of the International Conference on Machine Learning, 97, 1997, pp. 125–133.
[13] S. Wang, J. Wang, Z. Wang, Q. Ji, Enhancing multi-label classification by modeling dependencies among labels, Pattern Recognit. 47 (10) (2014) 3405–3413.
[14] J. Hernández-González, I. Inza, J.A. Lozano, Learning Bayesian network classifiers from label proportions, Pattern Recognit. 46 (12) (2013) 3425–3440.
[15] J. Peña, J. Lozano, P. Larrañaga, An improved Bayesian structural EM algorithm for learning Bayesian networks for clustering, Pattern Recognit. Lett. 21 (8) (2000) 779–786.
[16] S. Luengo-Sanchez, C. Bielza, P. Larrañaga, Hybrid Gaussian and von Mises model-based clustering, in: Proceedings of the 22nd European Conference on Artificial Intelligence (ECAI), IOS Press, 2016, pp. 855–862.
[17] G.F. Cooper, The computational complexity of probabilistic inference using Bayesian belief networks, Artif. Intell. 42 (2) (1990) 393–405.
[18] F.R. Bach, M.I. Jordan, Thin junction trees, in: Advances in Neural Information Processing Systems, 2001, pp. 569–576.
[19] G. Elidan, S. Gould, Learning bounded treewidth Bayesian networks, in: Advances in Neural Information Processing Systems, 2009, pp. 417–424.
[20] S. Nie, C.P. de Campos, Q. Ji, Efficient learning of Bayesian networks with bounded tree-width, Int. J. Approximate Reason. 80 (2017) 412–427.
[21] M. Scanagatta, G. Corani, M. Zaffalon, J. Yoo, U. Kang, Efficient learning of bounded-treewidth Bayesian networks from complete and incomplete data sets, Int. J. Approximate Reason. 95 (2018) 152–166.
[22] R.D. Shachter, Evidence absorption and propagation through evidence reversals, in: Proceedings of the 5th Annual Conference on Uncertainty in Artificial Intelligence, North-Holland Publishing Co., 1990, pp. 173–190.
[23] N.L. Zhang, D. Poole, A simple approach to Bayesian network computations, in: Proceedings of the 10th Canadian Conference on Artificial Intelligence, 1994, pp. 171–178.
[24] J. Pearl, A constraint propagation approach to probabilistic reasoning, in: Proceedings of the 1st Annual Conference on Uncertainty in Artificial Intelligence, 1985, pp. 357–369.
[25] A. Darwiche, Recursive conditioning, Artif Intell. 126 (1) (2001) 5–41.
[26] P.P. Shenoy, G. Shafer, Axioms for probability and belief-function propagation, in: Proceedings of the 4th Annual Conference on Uncertainty in Artificial Intelligence, North-Holland Publishing Co., 1990, pp. 169–198.
[27] F. Jensen, S. Lauritzen, K. Olsen, Bayesian updating in recursive graphical models by local computation, Comput. Stat. Quarter. 4 (1990) 269–282.
[28] R. Dechter, Bucket elimination: a unifying framework for reasoning, Artif. Intell. 113 (1) (1999) 41–85.
[29] A. Darwiche, Modeling and Reasoning with Bayesian Networks, Cambridge University Press, Cambridge, England, 2009.
[30] N. Robertson, P. Seymour, Graph minors. II. Algorithmic aspects of tree-width, J. Alg. 7 (3) (1986) 309–322.
[31] J. Kwisthout, H. Bodlaender, L. van der Gaag, The necessity of bounded treewidth for efficient inference in Bayesian networks, in: Proceedings of the 19th European Conference on Artificial Intelligence, IOS Press, 2010, pp. 237–242.
[32] J. Kwisthout, Most probable explanations in Bayesian networks: complexity and tractability, Int. J. Approximate Reason. 52 (9) (2011) 1452–1469.
[33] B.K. Sy, Reasoning MPE to multiply connected belief networks using message passing, in: Proceedings of the 10th National Conference on Artificial Intelligence, AAAI Press, 1992, pp. 570–576.
[34] J.D. Park, MAP complexity results and approximation methods, in: Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers Inc., 2002, pp. 388–396.
[35] S. Arnborg, D.G. Corneil, A. Proskurowski, Complexity of finding embeddings in a k-tree, SIAM J. Algebr. Discrete Methods 8 (2) (1987) 277–284.
[36] F.V. Fomin, Y. Villanger, Treewidth computation and extremal combinatorics, Combinatorica 32 (3) (2012) 289–308.
[37] H.L. Bodlaender, F.V. Fomin, A.M. Koster, D. Kratsch, D.M. Thilikos, On Exact Algorithms for Treewidth, Springer, New York City, NY, USA, 2006.
[38] H.L. Bodlaender, A.M. Koster, Treewidth computations i. upper bounds, Inf. Comput. 208 (3) (2010) 259–275.
[39] H.M. Markowitz, The elimination form of the inverse and its application to linear programming, Manage. Sci. 3 (3) (1957) 255–269.
[40] U.B. Kjærulff, Triangulation of Graphs-Algorithms giving small total state space, Technical Report, R-90-09, Department of Mathematics and Computer Science, Aalborg University, Denmark, 1990.
[41] D.J. Rose, R.E. Tarjan, G.S. Lueker, Algorithmic aspects of vertex elimination on graphs, SIAM J. Comput. 5 (2) (1976) 266–283.
[42] R.E. Tarjan, M. Yannakakis, Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs, SIAM J. Comput. 13 (3) (1984) 566–579.
[43] U.B. Kjærulff, Optimal decomposition of probabilistic networks by simulated annealing, Stat. Comput. 2 (1) (1992) 7–17.
[44] P. Larrañaga, C.M. Kuijpers, M. Poza, R.H. Murga, Decomposing Bayesian networks: triangulation of the moral graph with genetic algorithms, Stat. Comput. 7 (1) (1997) 19–34.
[45] H. Akaike, A new look at the statistical model identification, IEEE Trans. Automat. Contr. 19 (6) (1974) 716–723.
[46] G. Schwarz, Estimating the dimension of a model, Annal. Stat. 6 (2) (1978) 461–464.
[47] D. Heckerman, D. Geiger, D.M. Chickering, Learning Bayesian networks: the combination of knowledge and statistical data, Mach. Learn. 20 (3) (1995) 197–243.
[48] G.F. Cooper, E. Herskovits, A Bayesian method for the induction of probabilistic networks from data, Mach. Learn. 9 (4) (1992) 309–347.
[49] T. Wang, J.W. Touchman, G. Xue, Applying two-level simulated annealing on Bayesian structure learning to infer genetic networks, in: Proceedings of the 2004 IEEE Computational Systems Bioinformatics Conference, IEEE Computer Society, 2004, pp. 647–648.
[50] I. Tsamardinos, L. Brown, C. Aliferis, The max-min hill-climbing Bayesian network structure learning algorithm, Mach. Learn. 65 (1) (2006) 31–78.
[51] J.A. Gámez, J.L. Mateo, J.M. Puerta, Learning Bayesian networks by hill climbing: efficient methods based on progressive restriction of the neighborhood, Data Min. Knowl. Discov. 22 (1–2) (2011) 106–148.
[52] M. Ramoni, P. Sebastiani, Learning Bayesian networks from incomplete databases, in: Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence (UAI'97), Morgan Kaufmann Publishers, 1997, pp. 401–408.
[53] S.L. Lauritzen, The EM algorithm for graphical association models with missing data, Comput. Stat. Data Anal. 19 (2) (1995) 191–201.
[54] P.M. Rancoita, M. Zaffalon, E. Zucca, F. Bertoni, C.P. de Campos, Bayesian network data imputation with application to survival tree analysis, Comput. Stat. Data Anal. 93 (2016) 373–387.
[55] M. Benjumeda, C. Bielza, P. Larrañaga, Learning tractable Bayesian networks in the space of elimination orders, Artif. Intell. in Press (2018).
[56] K. Grant, M.C. Horsch, Methods for constructing balanced elimination trees and other recursive decompositions, Int. J. Approx. Reason. 50 (9) (2009) 1416–1424.
[57] S. Holm, A simple sequentially rejective multiple test procedure, Scand. J. Stat. (1979) 65–70.
[58] J.P. Shaffer, Modified sequentially rejective multiple test procedures, J. Am. Stat. Assoc. 81 (395) (1986) 826–831.
[59] S. Garcia, F. Herrera, An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons, J. Mach. Learn. Res. 9 (2008) 2677–2694.
[60] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[61] D. Lowd, J. Davis, Learning Markov network structure with decision trees, in: IEEE International Conference on Data Mining, IEEE, 2010, pp. 334–343.
[62] J. Van Haaren, J. Davis, Markov network structure learning: a randomized feature generation approach, in: Proceedings of the 26th AAAI Conference on Artificial Intelligence, 2012, pp. 1148–1154.
[63] J. Bekker, J. Davis, A. Choi, A. Darwiche, G.V.d. Broeck, Tractable learning for complex probability queries, in: Proceedings of the 28th International Conference on Neural Information Processing Systems, MIT Press, 2015, pp. 2242–2250.
[64] N. Friedman, The Bayesian structural EM algorithm, in: Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI-98), Morgan Kaufmann Publishers, 1998, pp. 129–138.
[65] L.K. Poon, N.L. Zhang, T. Liu, A.H. Liu, Model-based clustering of high-dimensional data: variable selection versus facet determination, Int. J. Approx. Reason. 54 (1) (2013) 196–215.
[66] O. Keivani, J.M. Peña, Uni-and Multi-dimensional clustering via Bayesian networks, in: Unsupervised Learning Algorithms, Springer, 2016, pp. 163–192.
[67] N. Zhang, Hierarchical latent class models for cluster analysis, J. Mach. Learn. Res. 5 (6) (2004) 697–723.
[68] H. Poon, P. Domingos, Sum-product networks: a new deep architecture, in: Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence, AUAI Press, 2011, pp. 337–346.

Marco Benjumeda recieved his M.Sc. degree in Computer Science from the Autonomous University of Madrid in 2012. He obtained a M.Sc. degree in Artificial Intelligence from the Technical University of Madrid (UPM) in 2014 and is currently a Ph.D. student at UPM's Artificial Intelligence Department and a member of the Computational Intelligence Group.

Sergio Luengo recieved his B.Sc. degree in Computer Science from the Alcalá de Henares University in 2013. He obtained a M.Sc. degree in Artificial Intelligence from the Technical University of Madrid (UPM) in 2014. He is currently a Ph.D. student at the UPM's Artificial Intelligence Department and a member of the Computational Intelligence Group, taking part in the Cajal Blue Brain project. He also collaborates in the Human Brain Project developing software modules.

Pedro Larranaga is Full Professor in Computer Science and Artificial Intelligence at the Technical University of Madrid (UPM) since 2007. He received the MSc degree in mathematics (statistics) from the University of Valladolid and the PhD degree in computer science from the University of the Basque Country ("excellence award"). Before moving to UPM, his academic career has been developed at the University of the Basque Country (UPV-EHU) at several faculty ranks: Assistant Professor (1985–1998), Associate Professor (1998–2004) and Full Professor (2004–2007). He earned the habilitation qualification for Full Professor in 2003. He has published more than 200 papers in impact factor journals and has supervised 25 PhD theses. He is fellow of the European Association for Artificial Intelligence since 2012. He has been awared the 2013 Spanish National Prize in Computer Science and the prize of the Spanish Association for Artificial Intelligence in 2018.

Concha Bielza received the M.S. degree in Mathematics from Universidad Complutense de Madrid, Madrid, Spain, in 1989 and the Ph.D. degree in Computer Science from Universidad Politécnica de Madrid, Madrid, in 1996 (extraordinary doctorate award). She is currently (since 2010) a Full Professor of Statistics and Operations Research with the Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid. She has published more than 100 papers in impact factor journals and has supervised 9 PhD theses. She was awared the 2014 UPM Research Prize.