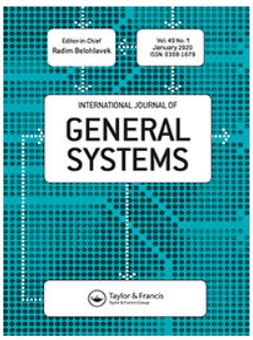


International Journal of General Systems

ISSN: 0308-1079 (Print) 1563-5104 (Online) Journal homepage: https://www.tandfonline.com/loi/ggen20

Learning bipartite Bayesian networks under monotonicity restrictions

Martin Plajner & Jiří Vomlel

To cite this article: Martin Plajner & Jiří Vomlel (2020) Learning bipartite Bayesian networks under monotonicity restrictions, International Journal of General Systems, 49:1, 88-111, DOI: 10.1080/03081079.2019.1692004

To link to this article: https://doi.org/10.1080/03081079.2019.1692004

INTERNATIONAL JOURNAL OF GENERAL SYSTEMS
2020, VOL. 49, NO. 1, 88–111
https://doi.org/10.1080/03081079.2019.1692004

Taylor & Francis
Taylor & Francis Group

Learning bipartite Bayesian networks under monotonicity restrictions

Martin Plajner [a,b] and Jiří Vomlel [b]

[a]Faculty of Nuclear Sciences and Physical Engineering, Czech Technical University, Prague, Czech Republic; [b]Institute of Information Theory and Automation, Czech Academy of Sciences, Prague 8, Czech Republic

ABSTRACT

Learning parameters of a probabilistic model is a necessary step in machine learning tasks. We present a method to improve learning from small datasets by using monotonicity conditions. Monotonicity simplifies the learning and it is often required by users. We present an algorithm for Bayesian Networks parameter learning. The algorithm and monotonicity conditions are described, and it is shown that with the monotonicity conditions we can better fit underlying data. Our algorithm is tested on artificial and empiric datasets. We use different methods satisfying monotonicity conditions: the proposed gradient descent, isotonic regression EM, and non-linear optimization. We also provide results of unrestricted EM and gradient descent methods. Learned models are compared with respect to their ability to fit data in terms of log-likelihood and their fit of parameters of the generating model. Our proposed method outperforms other methods for small sets, and provides better or comparable results for larger sets.

ARTICLE HISTORY
Received 15 December 2018
Accepted 30 October 2019

KEYWORDS
Bayesian networks; monotonicity; parameter learning; isotonic regression; gradient method; computerized adaptive testing

1. Introduction

Our research is focused in the domain of Computerized Adaptive Testing (CAT) working with Bayesian Networks (BNs) to model students' abilities, which is also addressed, for example, by Almond and Mislevy (1999), van der Linden and Glas (2000). CAT is a concept of testing latent student abilities, which allows creating shorter tests, asking fewer questions while keeping the same level of information. This task is performed by asking each individual student the right questions. Questions are selected based on a student model. In common practice, experts often use Item Response Theory models (IRT) (Rasch 1960), which are well explored and have been in use for a long time. Nevertheless, we have focused our attention on a different family of models. The reason is that Bayesian Networks provide us with better relationships in the model. It is, for example, possible to model more complex influences between skills and questions because BNs are not limited to connecting each skill with each question; moreover, we can introduce relationships between skills themselves. We address the topic of the model selection in larger detail in our previous work, e.g. Plajner and Vomlel (2016b).

CONTACT Martin Plajner martin.plajner@fjfi.cvut.cz

During our research, we have noticed that there are certain conditions which should be satisfied in this specific modeling task. We have especially been focused on monotonicity conditions. Monotonicity conditions incorporate qualitative influences into a model. These influences restrict conditional probabilities inside the model in a specific way to avoid unwanted behavior. Monotonicity in Bayesian Networks has been discussed in the literature for a long time. The most relevant papers are (Wellman 1990; Druzdzel and Henrion 1993) and more recently (Restificar and Dietterich 2013; Masegosa, Feelders, and der Gaag 2016). Monotonicity restrictions are often motivated by reasonable demands from model users. In our case of CAT, it means we want to guarantee that students having a higher level of skill(s) will have a higher probability of answering questions correctly. As another example of monotonicity usage, imagine a BN that is learned to predict the effect of commercial promotions of products in retail stores. There are certain factors which should have an isotone effect. For example, secondary placement in the store, i.e. the position in the store's layout. A better position should provide a better result. If it does not, it is most likely caused by other factors or noise in the data. In this case, we want the learned effect to be isotone and our proposed algorithm can be used to provide it.

Certain types of models include monotonicity naturally, due to the way in which they are constructed. This is not true in the case of general BNs. In order to satisfy these conditions, we have to introduce restrictions to conditional probabilities during the process of parameter learning.

In our previous work, we showed that monotonicity conditions are useful in the context of CAT (Plajner and Vomlel 2016b). Later we applied these conditions to Bayesian Networks (Plajner and Vomlel 2017). In this article, we present a gradient descent optimum search method for BN parameter learning under monotonicity conditions. The algorithm we present provides a tool to include monotonicity in the BN models with multiple-state variables. We implemented the new method in R language and performed experimental verification of our assumptions. Experiments were performed on two datasets. The first one, a synthetic dataset, is generated from artificial models satisfying monotonicity conditions. The second one, an empirical dataset, is newly obtained and it consists of data from the Czech high school state final exam. This second dataset contains a large volume of reliable data, and it is very useful for the empirical verification of our approach. Experiments on these datasets were performed with various parameter learning methods both satisfying and not satisfying the monotonicity restrictions. The results are compared to show differences between individual methods and the approaches with and without considering monotonicity.

In contrast to our previously published articles, this paper brings significant modifications and improvements. Here, we establish a way of using our proposed gradient descent algorithm for BNs that have other than binary variables. We also modified the irEM method, which we use as a reference for the work with multi-state variables. In this article, we add a new dataset, which is based on large scale real-world data in a domain where the monotonicity should apply. Moreover, we have revised the way to evaluate models in order to create a more precise and comprehensive evaluation. This step includes adding to the comparison of additional monotonicity-ensuring methods.

The structure of this article is as follows. First, we establish our notation and describe monotonicity conditions in detail in Section 2.1. Next, we present different methods for learning parameters under monotonicity conditions in Section 3 and afterwords we present

our proposed method in Section 3.1. In Section 4, we take a closer look at the experimental setup and present results of our experiments. Section 5 contains an overview and a discussion of the obtained results.

2. BN models and monotonicity

2.1. Models and adaptive testing

In our work we focus on computerized adaptive testing and assessing student knowledge and abilities, using Bayesian Networks with a specific structure. The structure is a bipartite network, which consists of a layer of skills and a layer of questions. Skills are parents in our structure and correspond to specific abilities a student may or may not have. Individual states of these skills are interpreted as levels of knowledge. This interpretation is generally difficult as skills are unobserved variables. Having monotonicity constraints in our models, we are able to introduce an ordering of these levels and refer to them as increasing (or decreasing) qualities of skills. Children in the bipartite structure are question nodes, which correspond to particular questions in a test. Levels of these nodes correspond to the points obtained by solving the specific problem (the problem can be divided into subproblems with different scores). These models are described in further detail in Plajner and Vomlel (2016a).

2.2. Notation

In this article, we use BNs to model students in the domain of CAT. Details about BNs can be found, for example, in Pearl (1988), Nielsen and Jensen (2007). The model we use can be considered a special BN structure such as Multi-dimensional Bayesian Network Classifier which is described, e.g. in van der Gaag and Waal (2006). We restrict ourselves to the BNs that have two levels. In compliance with our previous articles, variables in the parent level are skill variables S. The child level contains question variables X. Examples of network structures, which we also used for experiments, are shown in Figures 1 and 2.

- We use the symbol $\boldsymbol{X}$ to denote the multivariable $(X_1, \ldots, X_n)$ taking states $\boldsymbol{x} = (x_1, \ldots, x_n)$. The total number of question variables is n, the set of all indices of question variables is $\boldsymbol{N} = \{1, \ldots, n\}$. Question variables' individual states are $x_{i,t}, t \in \{0, \ldots, n_i\}$ and they are observable. Each question can have a different number of states; the maximum number of states over all variables is $N^{\max} = \max_i(n_i) + 1$. States are integers with the natural ordering.[1]

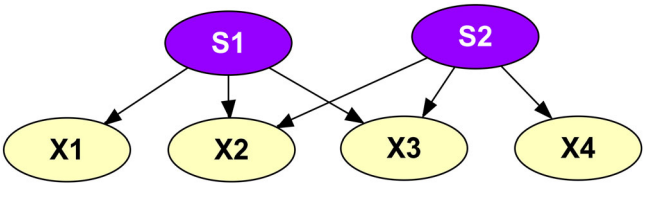


Figure 1. An artificial BN model.

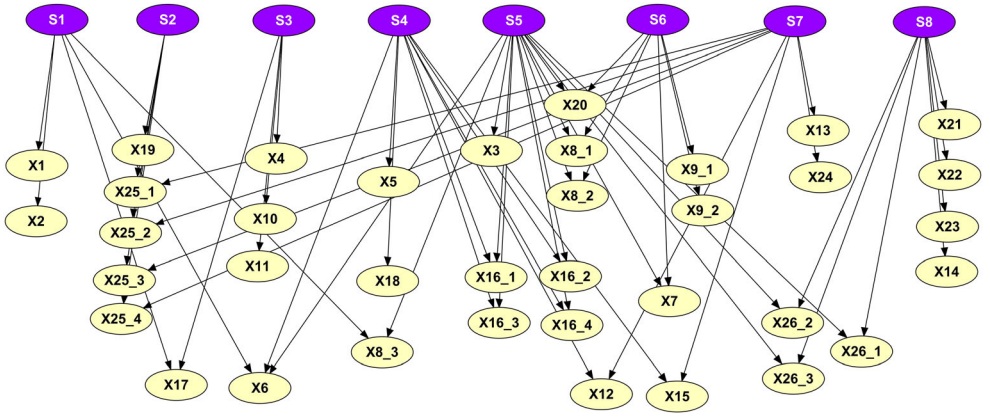


Figure 2. A BN model for CAT.

- We use the symbol $\boldsymbol{S}$ to denote the multivariable $(S_1, \ldots, S_m)$ taking states $\boldsymbol{s} = (s_1, \ldots, s_m)$. The set of all indices of skill variables is $\boldsymbol{M} = \{1, \ldots, m\}$. Skill variables have variable numbers of states, the number of states of a variable S_j is m_j, and individual states are $s_{j,k}, k \in \{1, \ldots, m_j\}$. The variable $\boldsymbol{S}^i = \boldsymbol{S}^{pa(i)}$ stands for a multivariable containing the parent variables of the question X_i. Indices of these variables are $\boldsymbol{M}^i \subseteq \boldsymbol{M}$. The set of all possible state configurations of $\boldsymbol{S}^i$ is $Val(\boldsymbol{S}^i)$. Skill variables are unobservable.

The BN is defined by, along with its structure, parameters of all questions $X_i, i \in \boldsymbol{N}, \boldsymbol{s}^i \in Val(\boldsymbol{S}^i)$ which define conditional probabilities as

$$\theta^t_{i,\boldsymbol{s}^i} = P(X_i = t | \boldsymbol{S}^i = \boldsymbol{s}^i),$$

and, parameters of all skills $S_j, j \in \boldsymbol{M}$ as

$$\tilde{\theta}_{j,s_j} = P(S_j = s_j).$$

From the definition above it follows that the parameters are constrained to be between zero and one with constraints for question variables $\sum_t \theta^t_{i,\boldsymbol{s}^i} = 1, \ \forall i, \boldsymbol{s}^i$ and, for parent variables, $\sum_{s_j} \tilde{\theta}_{j,s_j} = 1, \ \forall j$. To avoid these constraints in our gradient method, we reparametrize

$$\theta^t_{i,\boldsymbol{s}^i} = \frac{\exp(\mu^t_{i,\boldsymbol{s}^i})}{\sum_{t'=0}^{n_i} \exp(\mu^{t'}_{i,\boldsymbol{s}^i})},$$

$$\tilde{\theta}_{j,s_j} = \frac{\exp(\tilde{\mu}_{j,s_j})}{\sum_{s'_j=1}^{m_i} \exp(\tilde{\mu}_{j,s'_j})}.$$

The set of all question parameters $\theta^t_{i,\boldsymbol{s}^i}$ and all skills parameters $\tilde{\theta}_{j,s_j}$ is denoted by $\boldsymbol{\theta}$ and $\boldsymbol{\mu}$ is the set of reparameterized parameters. The symbol $\boldsymbol{\mu_{i,s^i}} = \{\mu^{t'}_{i,\boldsymbol{s}^i}, t' \in \{0, \ldots, n_i\}\}$ stands for the set of parameters for all states of a question X_i given one parent configuration $\boldsymbol{s}^i$. Theoretically, $\mu^{t'}_{i,\boldsymbol{s}^i} \in \mathbb{R}, \forall i, \forall t'$ but for the practical computational issues we forbid the two extreme values of $\boldsymbol{\theta}$, i.e. 0 and 1. We elaborate more on exact bounds in the experimental section of this paper in Section 4.

2.3. Monotonicity

The concept of monotonicity in BNs has been discussed in the literature since the 1990s, see Wellman (1990), Druzdzel and Henrion (1993). Later, its benefits for BN parameter learning were addressed, for example, by van der Gaag, Bodlaender, and Feelders (2004), Altendorf, Restificar, and Dietterich (2005), Feelders and der Gaag (2005). This topic is still active, see, e.g. Restificar and Dietterich (2013), Masegosa, Feelders, and der Gaag (2016).

We consider only variables with states from $\mathbb{N}_0$ with their natural ordering, i.e. the ordering of states of skill variable S_j for $j \in \boldsymbol{M}$ is

$$s_{j,1} \prec \cdots \prec s_{j,m_j}.$$

A variable S_j has an *isotone effect* on its child X_i if for all $k, l \in \{1, \ldots, m_j\}, t' \in \{0, \ldots, n_i - 1\}$ the following holds:[2]

$$s_{j,k} \preceq s_{j,l} \Rightarrow \sum_{t=0}^{t'} P(X_i = t | S_j = s_{j,k}, \boldsymbol{s}) \;\geq\; \sum_{t=0}^{t'} P(X_i = t | S_j = s_{j,l}, \boldsymbol{s}),$$

and *antitone effect*:

$$s_{j,k} \preceq s_{j,l} \Rightarrow \sum_{t=0}^{t'} P(X_i = t | S_j = s_{j,k}, \boldsymbol{s}) \;\leq\; \sum_{t=0}^{t'} P(X_i = t | S_j = s_{j,l}, \boldsymbol{s}),$$

where $\boldsymbol{s}$ is a configuration of the remaining parents of question i without S_j. For each question $X_i, i \in \boldsymbol{M}$ we denote by $\boldsymbol{S}^{i,+}$ the set of parents with an isotone effect and by $\boldsymbol{S}^{i,-}$ the set of parents with an antitone effect.

The conditions above are defined for the states of question variable X_i in the set $\{0, \ldots, n_i - 1\}$. The sum property of conditional probabilities

$$\sum_{t=0}^{n_i} \theta^{t}_{i,\boldsymbol{s}^i} = 1,$$

implies that, for n_i in the case of the isotone effect:

$$s_{j,k} \preceq s_{j,l} \Rightarrow P(X_i = n_i | S_j = s_{j,k}, \boldsymbol{s}) \;\leq\; P(X_i = n_i | S_j = s_{j,l}, \boldsymbol{s}),$$

and in the case the antitone effect:

$$s_{j,k} \preceq s_{j,l} \Rightarrow P(X_i = n_i | S_j = s_{j,k}, \boldsymbol{s}) \;\geq\; P(X_i = n_i | S_j = s_{j,l}, \boldsymbol{s}).$$

Next, we define a partial ordering $\preceq_i$ on all state configurations of parents $\boldsymbol{S}^i$ of the i-th question, if for all $\boldsymbol{s}^i, \boldsymbol{r}^i \in Val(\boldsymbol{S}^i)$:

$$\boldsymbol{s}^i \preceq_i \boldsymbol{r}^i \Leftrightarrow \left(s^i_j \preceq r^i_j,\ j \in \boldsymbol{S}^{i,+}\right) \quad \text{and} \quad \left(r^i_j \preceq s^i_j,\ j \in \boldsymbol{S}^{i,-}\right).$$

The monotonicity condition requires that the probability of an incorrect answer is higher for a lower order parent configuration (the chance of a correct answer increases for higher

ordered parents' states), i.e. for all $\boldsymbol{s}^i, \boldsymbol{r}^i \in Val(\boldsymbol{S}^i), k \in \{0, \ldots, n_i - 1\}$:

$$\boldsymbol{s}^i \preceq_i \boldsymbol{r}^i \Rightarrow \sum_{t=0}^{k} P(X_i = t | \boldsymbol{S}^i = \boldsymbol{s}^i) \geq \sum_{t=0}^{k} P(X_i = t | \boldsymbol{S}^i = \boldsymbol{r}^i).$$

In our experimental part, we consider only the isotone effect of parents on their children. The difference with antitone effects is only in the partial ordering.

3. Learning model parameters under monotonicity conditions

Different methods can be used to learn model parameters while satisfying monotonicity conditions. In this section, we will outline some of them and then we will describe our newly proposed method. All optimization methods we consider are optimizing the log-likelihood of the model. Methods, in the order as they are described below, are:

- isotonic regression EM (**irEM**),
- bounded non-linear optimization (**Cobyla**),
- restricted gradient method (**res gradient**).

Isotonic regression EM

Isotonic Regression EM was proposed in Masegosa, Feelders, and der Gaag (2016). The authors propose a method for parameter learning which ensures convergence to monotonicity satisfying parameters. The method is a modification of the well-known EM algorithm where the M-step is modified to contain an isotonic regression step. This step, in the case of a solution not complying with the monotonicity conditions, moves the solution to the border of the admissible parameter space. The steps of the algorithm are applied iteratively as in the case of the regular EM. In each step a new solution, starting from the previous point, is found. This solution may or may not satisfy the monotonicity conditions. If it does not, isotonic regression is performed to satisfy them. As we show later in this paper, this behavior has a tendency to end at the border of the admissible parameter space. This behavior may imply that the algorithm fails to provide an optimal solution.

We have implemented the generalized version of the irEM algorithm working with multiple-state parent variables in our previous paper (Plajner and Vomlel 2017). In the present paper, we further generalize the irEM method to work with multiple states of children variables as well.

The authors of the irEM algorithm also provide quick-irEM, abbreviated to qirEM, a version of the algorithm which is a speed optimization modification. In this case, the isotonic regression step is performed only once after the EM algorithm converges. In experiments, we have tested this version of the algorithm as well.

Bounded non-linear optimization

The monotonicity constraints form a subspace in the whole parameter space of CPTs' parameters. A possible approach is to apply an optimization method for finding an optimum only inside this subspace. In that case, the solution would satisfy the monotonicity

constraints and should be optimal (locally or globally based on the algorithm and properties of the space itself). In our experiments, we used various methods from NLOPT library for non-linear optimization problems (Johnson 2018). From among methods available in the library, we selected Sequential Least-Squares Quadratic Programming (Kraft 1994) and Constrained Optimization BY Linear Approximations (Cobyla) (Powell 1994) methods. Reasons to select these methods are that they are able to work in our domain of restricted space and non-linear inequalities formed by the monotonicity constraints. They are local optimization techniques and as such they do not guarantee global optimum. We have also experimented with global optimization methods but the time required for these methods to converge was extensive and this is why we decided to skip experiments with these methods.

Restricted gradient method

We propose to use the Restricted Gradient Search method (which is our proposed method) to find parameters of a BN under monotonicity restrictions. This method uses the gradient descent optimum search technique. It takes a penalized log-likelihood function to be optimized in order to find the solution of this problem. The penalization encourages the solution to leave the non-admissible area of non-monotonic parameters and leads the gradient towards a monotonic solution. As such, this method does not strictly ensure monotonicity to hold. Nevertheless, there are two important comments to be made. The strength of the restriction is variable and setting high restriction values effectively enforces the solution to be monotonic. Moreover, if the solution is not monotonic the reason might be that the underlying data strongly contradicts it. This method provides an option to balance data evidence and the monotonicity restrictions and allows to create a non-monotonic solution. Even though this is possible to achieve, there is no general rule on how to weight these influences. It depends on the data and the model and requires expertise to evaluate. If the user is not sure, we propose to use large penalty values to practically ensure a monotonic solution. This method is described in detail in the following Section.

3.1. Parameter gradient search with monotonicity

We have developed a method based on gradient descent optimization. We follow the work of Altendorf, Restificar, and Dietterich (2005) where the authors use a gradient descent method with exterior penalties. The main difference is that we consider models with hidden variables. In this article, we generalize the method from Plajner and Vomlel (2017) to multi-state question variables.

We denote by $\boldsymbol{D}$ the set of indices of the question vectors. One vector $x^k, k \in \boldsymbol{D}$ corresponds to one student and an observation of i-th variable X_i is x_i^k. The number of occurrences of the k-th configuration vector in the data sample is d_k.

We use the BN model described in Section 2.1 where we have unobserved parent variables and observed children variables. The parent variables correspond to skills and the number of their levels set the levels of quality/ability of the skill. The child nodes correspond to questions and the number of levels is the number of possible points obtained in the particular question. Let $\boldsymbol{I}_t^k, t \in \{0, \ldots, N^{\max}\}$ be sets of indices of the questions in a state

t. Then, we define the following products based on the observations in the k-th vector:[3]

$$p^t(\boldsymbol{\mu}, \boldsymbol{s}, k) = \begin{cases} 1, & \text{if } \boldsymbol{I}_t^k = \emptyset \\ \prod_{i \in \boldsymbol{I}_t^k} \frac{\exp(\mu^t_{i,\boldsymbol{s}^i})}{\sum_{t'=0}^{n_i} \exp(\mu^{t'}_{i,\boldsymbol{s}^i})}, & \text{otherwise} \end{cases}, \quad t \in \{0, \ldots, N^{\max}\},$$

$$p_\mu(\boldsymbol{\mu}, \boldsymbol{s}, k) = \prod_{t=0}^{N^{\max}} p^t(\boldsymbol{\mu}, \boldsymbol{s}, k),$$

$$p_{\tilde{\mu}}(\boldsymbol{\mu}, \boldsymbol{s}) = \prod_{j=1}^{m} \exp(\tilde{\mu}_{j,s_j}).$$

We work with the log-likelihood of data modeled by BN with the parameter vector $\boldsymbol{\mu}$:

$$\begin{aligned} LL(\boldsymbol{\mu}) &= \sum_{k \in \boldsymbol{D}} d_k \cdot \log \left(\sum_{\boldsymbol{s} \in \mathrm{Val}(\boldsymbol{S})} \prod_{j=1}^{m} \frac{\exp(\tilde{\mu}_{j,s_j})}{\sum_{s'_j=1}^{m_j} \exp(\tilde{\mu}_{j,s'_j})} \cdot p_\mu(\boldsymbol{\mu}, \boldsymbol{s}, k) \right) \\ &= \sum_{k \in \boldsymbol{D}} d_k \cdot \log \left(\sum_{\boldsymbol{s} \in \mathrm{Val}(\boldsymbol{S})} p_{\tilde{\mu}}(\boldsymbol{\mu}, \boldsymbol{s}) p_\mu(\boldsymbol{\mu}, \boldsymbol{s}, k) \right) - N \cdot \sum_{j=1}^{m} \log \sum_{s'_j=1}^{m_j} \exp(\tilde{\mu}_{j,s'_j}). \end{aligned}$$

Monotonicity restrictions for the gradient search

To enforce monotonicity into the model we apply a penalty function which penalizes solutions that do not satisfy the monotonicity conditions. We will use the following penalization function for the log-likelihood:

$$C(\boldsymbol{\mu}_{\boldsymbol{i},\boldsymbol{s}^i}, \boldsymbol{\mu}_{\boldsymbol{i},\boldsymbol{r}^i}, \hat{t}, c) = \max \left(0, c \cdot \left(\frac{\sum_{t=0}^{\hat{t}} \exp(\mu^t_{i,\boldsymbol{r}^i})}{\sum_{t'=0}^{n_i} \exp(\mu^{t'}_{i,\boldsymbol{r}^i})} - \frac{\sum_{t=0}^{\hat{t}} \exp(\mu^t_{i,\boldsymbol{s}^i})}{\sum_{t'=0}^{n_i} \exp(\mu^{t'}_{i,\boldsymbol{s}^i})} \right)^p \right),$$

and the penalized log-likelihood is

$$LL'(\boldsymbol{\mu}, c) = LL(\boldsymbol{\mu}) - \sum_{i \in \boldsymbol{N}} \sum_{\boldsymbol{s}^i \preceq_i \boldsymbol{r}^i} \sum_{\hat{t}=0}^{N^{\max}-1} C\left(\boldsymbol{\mu}_{\boldsymbol{i},\boldsymbol{s}^i}, \boldsymbol{\mu}_{\boldsymbol{i},\boldsymbol{r}^i}, \hat{t}, c\right),$$

where p sets the degree of the polynomial function and it takes only odd values, c is a constant determining the slope of the penalization function, and $\hat{t}$ is the level of the question node. The higher the value of c the more strict the penalization is. Theoretically, this condition does not ensure monotonicity but, practically, selecting high values of c results in monotonic estimates. The polynomial penalty uses an odd degree polynomial function. We discuss the size of the penalty in the following Section.

Using the penalized log-likelihood, $LL'(\boldsymbol{\mu}, c)$, and its gradient $\nabla(LL'(\boldsymbol{\mu}, c))$, we can use standard gradient descent optimization methods to learn the parameter vector $\boldsymbol{\mu}$ of BN models. We provide formulas to compute the gradient in Appendix 1.

3.2. Ensuring monotonicity with the penalization

Penalization described above may provide a solution which is not monotone. This behavior is observable especially in instances in which the data strongly contradict the monotonicity conditions. The solution will always be close to the admissible region but the distance depends on the strength of the penalization. It depends on the specific application whether we require a strictly monotone result or not. In many cases, it may be acceptable to break these conditions in order to get a better data fit. When the training sample is very small, it is particularly easy to have data that contradicts monotonicity. However, in some situations, we need to enforce monotonicity. It is particularly easy to measure the distance from the border of the admissible region. We can use the iterative process to ensure monotonicity. If the final parameter vector after the optimization violates the monotonicity conditions, we restart the optimization with a stronger penalization and use the end point as a new starting point. This process is repeated until the monotone solution is reached. Nevertheless, in this section, we also provide a way to ensure monotonicity conditions by setting a strong enough penalization.

In order to be able to do that and to compare this method with other strictly monotone methods, we propose the following concept. Below we use the penalization $C(\boldsymbol{\mu}_{i,\boldsymbol{s}^i}, \boldsymbol{\mu}_{i,\boldsymbol{r}^i}, \hat{t}, c)$.

The penalization of the log-likelihood described above and detailed in Appendix 1 has to lead the gradient method to the admissible area. We need to ensure that for each $\mu^t_{i,\boldsymbol{s}^i}$ with the parent configuration $\boldsymbol{s}^i$ in the term

$$\frac{\partial LL'(\boldsymbol{\mu}, c)}{\partial \mu^t_{i,\boldsymbol{s}^i}} = \frac{\partial LL(\boldsymbol{\mu})}{\partial \mu^t_{i,\boldsymbol{s}^i}} \tag{1}$$

$$- \sum_{\boldsymbol{s}^i \preceq_i \boldsymbol{r}^i} \sum_{\hat{t}=0}^{N^{\max}} \frac{\partial C\left(\boldsymbol{\mu}_{i,\boldsymbol{s}^i}, \boldsymbol{\mu}_{i,\boldsymbol{r}^i}, \hat{t}, c\right)}{\partial \mu^t_{i,\boldsymbol{s}^i}} \tag{2}$$

$$- \sum_{\boldsymbol{r}^i \preceq_i \boldsymbol{s}^i} \sum_{\hat{t}=0}^{N^{\max}} \frac{\partial C\left(\boldsymbol{\mu}_{i,\boldsymbol{r}^i}, \boldsymbol{\mu}_{i,\boldsymbol{s}^i}, \hat{t}, c\right)}{\partial \mu^t_{i,\boldsymbol{s}^i}}, \tag{3}$$

the gradient part of $LL(\boldsymbol{\mu}, c)$ (1) is not larger than the penalization terms (2) and (3) while the parameter vector $\boldsymbol{\mu}$ is not in the admissible region. The two terms (2) and (3) of the penalization gradient are generated by the monotonicity conditions where each condition generates one item to the outer sum for one or both of them. The first term (2) is for the situation $\boldsymbol{s}^i \preceq_i \boldsymbol{r}^i$ and the second one (3) for the opposite instance $\boldsymbol{r}^i \preceq_i \boldsymbol{s}^i$. For a single parameter $\mu^t_{i,\boldsymbol{s}^i}$ these two gradient parts have opposite effects.

We need to analyze the partial ordering of skill configurations

$$\boldsymbol{r}^i \preceq_i \boldsymbol{s}^i, \boldsymbol{s}^i \preceq_i \boldsymbol{r}^i,\ \boldsymbol{s}^i, \boldsymbol{r}^i \in \mathrm{Val}(\boldsymbol{S}^i),$$

determining conditions of the question X_i and, more specifically, a single parameter $\mu^t_{i,\boldsymbol{s}^i}$. Because the penalty and its gradient is zero when the condition is not violated, we can omit the state configurations for which the condition holds and work only with the configurations for which the penalty is positive, i.e. all pairs $\boldsymbol{s}^i, \boldsymbol{r}^i \in \mathrm{Val}'(\boldsymbol{S}^i) \subseteq \mathrm{Val}(\boldsymbol{S}^i)$ for

which

$$C(\boldsymbol{\mu}_{\boldsymbol{i},\boldsymbol{s}^i}, \boldsymbol{\mu}_{\boldsymbol{i},\boldsymbol{r}^i}, \hat{t}, c) > 0.$$

Given this reduced set $Val'(\boldsymbol{S}^i)$ and the partial ordering, there is always one state configuration which is the first in the partial ordering. It means that there exists at least one configuration $\hat{\boldsymbol{s}}^i$ for which

$$\left\{\hat{\boldsymbol{s}}^i \preceq_i \boldsymbol{r}^i\right\} = \emptyset, \forall \boldsymbol{r}^i \in \mathrm{Val}'(\boldsymbol{S}^i).$$

In the part of the gradient corresponding to parameter $\mu^t_{i,\boldsymbol{s}^i}$ one of the two sums ($\boldsymbol{r}^i \preceq_i \hat{\boldsymbol{s}}^i$) is zero. If we are able to ensure that the penalization part of the gradient is always larger outside of the admissible region for this parameter, it will be moved to the admissible region by the gradient method. After this step, the whole solution either is in the admissible region, or we can use the same process to move another parameter to the admissible region as long as there are any parameters outside of the region.

The penalization drops towards zero as it gets closer to the border of the admissible region. This behavior creates computational difficulties. The cause of these difficulties lies in the fact that very small values of penalization can be outweighed by improvements of the log-likelihood by shifting parameters outside of the admissible space. Thus, it would be hard to ensure monotonicity in such conditions. To avoid these issues, we shrink the admissible region by adding a small margin β to the penalization function:

$$\begin{aligned} &C'_{2,p}(\boldsymbol{\mu}_{\boldsymbol{i},\boldsymbol{s}^i}, \boldsymbol{\mu}_{\boldsymbol{i},\boldsymbol{r}^i}, \hat{t}, c, \beta) \\ &\quad = \max\left(0, c \cdot \left(\frac{\sum_{t=0}^{\hat{t}} \exp(\mu^t_{i,\boldsymbol{r}^i})}{\sum_{t'=0}^{n_i} \exp(\mu^{t'}_{i,\boldsymbol{r}^i})} - \frac{\sum_{t=0}^{\hat{t}} \exp(\mu^t_{i,\boldsymbol{s}^i})}{\sum_{t'=0}^{n_i} \exp(\mu^{t'}_{i,\boldsymbol{s}^i})} + \beta\right)^p\right). \end{aligned}$$

This makes the lowest possible value at the border of the admissible region to be

$$C^*_{2,p}(\boldsymbol{\mu}_{\boldsymbol{i},\boldsymbol{s}^i}, \boldsymbol{\mu}_{\boldsymbol{i},\boldsymbol{r}^i}, \hat{t}, c, \beta) = c \cdot \beta^p.$$

As the penalization function is growing rapidly outside of the admissible region, it is sufficient to ensure the gradient inequality between terms (1), (2) and (3) in the formula above at the border.

Based on the reasoning above and the formulas to compute the gradient, we set the constant $c = c^*$ to ensure monotonicity in the following way

$$\begin{aligned} \left|\frac{\partial LL(\boldsymbol{\theta})}{\partial \theta^t_{i,\boldsymbol{s}^i}}\right| &< \left|\frac{\partial C\left(\boldsymbol{\theta}_{\boldsymbol{i},\boldsymbol{s}^i}, \boldsymbol{\theta}_{\boldsymbol{i},\boldsymbol{r}^i}, \hat{t}, c\right)}{\partial \theta^t_{i,\boldsymbol{s}^i}}\right| \\ \frac{1}{(\tilde{\theta}_-)^m \cdot (\theta_-)^n} &< cp\beta^{p-1} \cdot (\theta_-)^2 \\ c &> (\tilde{\theta}_-)^{-m} \cdot (\theta_-)^{-n-2} \cdot \beta^{1-p}, \end{aligned}$$

where $\beta = 0.01$ and $p = 3$ are chosen constants for the penalization. $\tilde{\theta}_-$ and θ_- are the minimal possible parameters values as

$$\tilde{\theta}_- = \frac{\exp(\tilde{\mu}_-)}{(m-1)\exp(\tilde{\mu}_+) + \exp(\tilde{\mu}_-)},$$
$$\theta_- = \frac{\exp(\mu_-)}{(n-1)\exp(\mu_+) + \exp(\mu_-)},$$

where $\tilde{\mu}_-, \tilde{\mu}_+, \mu_-, \mu_+$ are the bounds on reparameterized parameters which we use to prevent the probability values from reaching zero or one. In our case, we use the maximum of 3 and the minimum of -3, which effectively changes the interval of probabilities of a three-state variable to approximately $[0.0012; 0.995]$.

3.3. Isotonic regression EM for variables with multiple states

As mentioned earlier we use the isotonic regression EM method as a comparison method to our proposed gradient approach. Our algorithm is designed for variables having multiple states. The original irEM algorithm as it is published in Masegosa, Feelders, and der Gaag (2016) only works with binary variables. In our previous paper (Plajner and Vomlel 2017) we detailed our implementation of this method to work with parent variables in bipartite networks having multiple states. In order to be able to make the full comparison with the method proposed in this paper, we also implemented the irEM algorithm based on original work of Masegosa, Feelders, and der Gaag (2016), and Feelders (2007), where more information about the generalization to non-binary cases can be found, to work with multi-state child variables as well. We provide details of our implementation in this section.

For the sake of simplicity, we describe the implementation for isotone effects only as antitone effects are simple reversions. In our case of multiple states, a variable S_j has an *isotone effect* on its child X_i if for all $k, l \in \{1, \ldots, m_j\}, t' \in \{0, \ldots, n_i - 1\}$:

$$s_{j,k} \preceq s_{j,l} \Rightarrow \sum_{t=0}^{t'} P(X_i = t | S_j = s_{j,k}, \boldsymbol{s}) \geq \sum_{t=0}^{t'} P(X_i = t | S_j = s_{j,l}, \boldsymbol{s}).$$

More information about the stochastic dominance which is used here to model monotonicity in terms of cumulative distributions can be found in Wellman (1990).

The difference between the binary case and the multi-state case lies in the cumulative probability. In the binary case, there was only one series of inequalities for $t = 0$. Nevertheless, the structure of inequalities is the same as in the original irEM algorithm for each level of $t \in \{0, \ldots, n_i\}$. We propose the use of a series of isotonic regression steps. Each step works with a single level of t, i.e. with cumulative probabilities of question X_i in separate sets

$$I^t = \left\{ \sum_{t'=0}^{t} \theta_{i,\boldsymbol{s}^i}^{t'} \right\}, \forall t \in \{0, \ldots, n_i\}.$$

We perform the isotonic regression algorithm for each set separately with weights as relative frequencies in the same way as in the standard irEM algorithm to obtain new cumulative probabilities. These probabilities are afterward converted back to non-cumulative probabilities, i.e. individual variables.

4. Experiments

In experiments we would like to verify that if we learn parameters of BNs from a small volume of data it is beneficial to use monotonicity constraints. We designed two experiments to compare the methods discussed in this paper. In the first experiment we use artificial (synthetic) data; the other uses a real-world empiric data sample. There are two model versions for each dataset. One with binary and one with ternary question nodes, creating a total of four different model types we worked with.

Parameters are learned using the methods described above: our gradient method, unrestricted gradient descent, irEM, qirEM, regular EM, and Cobyla from NLOPT methods family .[4] For all model types, we learn the model parameters from subsets of data of different sizes. The quality of the parameter fit is measured by the log-likelihood of the learned models. The log-likelihood is measured on the whole dataset to provide results comparable between learning subsets of different sizes.

We implemented the methods in R and its various built-in packages to ease this process (R Development Core Team 2008), the NLOPT package mentioned above, and for computations of the regular EM algorithm, the Hugin (Hugin 2014) engine was used as the most time efficient tool. One important point to mention is that we restricted the parameters of the learned conditional probability tables to be from the specific interval $[\epsilon, 1-\epsilon]$ where $\epsilon \in [0,1]$ is a chosen small number; we used $\epsilon = 10^{-3}$. This step is carried out in order to avoid extreme parameter values. When the learning sample is very small, the networks parameters tend to move towards zero or one, but we know it should not be the case in reality. These limits are very similar for the reparameterized case of our gradient method as described in Section 3.2. The gradient method is penalized by the constraint described in Section 3.1 and it takes the parameter p defining the degree of the polynomial function. In our experiments we have always used the third degree as it proves, empirically, to converge fastest to the solution.

4.1. Artificial model

The structure of the first model is shown in Figure 1. This model has a typical model structure used in CAT where there are two levels of variables, one level of questions, and one level of skills (parents). Skills S_1 and S_2 have three possible states and questions X_1, X_2, X_3, X_4 are either binary or ternary, creating two different sets for further testing. Models were set up with 10 different sets of parameters $\boldsymbol{\theta}_a^*$ satisfying the monotonicity conditions. Furthermore, each model was used to generate one million of data samples (test results of a student, i.e. answers to questions). Parent variables were unobserved in all cases.

To learn the parameters of these models, we drew random subsets of size $d = 10^k$, where $k \in \{1,2,3,4,5,6\}$. Note that for $k = 6$ the subset is the set itself. Ten different sets for each size (indexed by b) were generated. Next, we created 10 initial starting points (indexed by c) for the model learning phase. The structures of both the generating and the learned models are fixed to be the same as it is shown in Figure 1. The starting parameter vectors $\boldsymbol{\mu}_b$ and the corresponding $\boldsymbol{\theta}_b$ were randomly generated from the interval $[0.01, 0.99]$. The starting points were the same for all methods. In this setup, we have 10 different original models, 10 different observation subsets, and 10 different starting parameters, which provides us with a thousand combinations for each set size and each model. Each model $M^d, d \in \{10^k, k \in$

$\{1, \ldots, 10\}\}$ is specified by a set of parameters $\boldsymbol{\theta}^d_{a,b,c}$, $a, b, c \in \{1, \ldots, 10\}$. We performed experiments for all these combinations and the results are evaluated as follows.

We measure the log-likelihood on the whole dataset in order to keep the results comparable. The log-likelihood of each learned model is compared with the log-likelihood of the generating model and then averaged over all instances of (a, b, c). This process gives us the average log-likelihood ratio between the generating and the fitted model for each subset of size d:

$$LR^d = \frac{1}{1000} \sum_{a,b,c} \frac{LL(\boldsymbol{\theta}^*_a)}{LL(\boldsymbol{\theta}^d_{a,b,c})}.$$

In this artificial setup, we are also able to measure the distance of the probability distributions of learned parameters Q from the distribution of generating parameters P. First, we calculate the average Kullback–Leibler divergence for each learned model:

$$D_{KL}(\theta^*_a || \theta^d_{a,b,c}) = \frac{1}{n} \sum_{i=1}^{n} D_{KL}(P(X_i|\theta^*_a) || Q(X_i|\theta^d_{a,b,c})).$$

Next we average over all results for each subset of size d:

$$D^d_{KL} = \frac{1}{1000} \sum_{a,b,c} D_{KL}(\theta^*_a || \theta^d_{a,b,c}).$$

We require all methods which are restricted to satisfy the monotonicity conditions. As described in Section 3.2, we can ensure this behavior by setting a high value of the penalization parameter c. In this setting $c = 10^{20}$ satisfies this condition. Even though it is possible to use this penalization, it is very high and in certain cases, it is numerically hard to reach convergence without the algorithm failing. Instead we use a smaller penalization of $c = 10^5$ which is, together with offset $\beta = 0.01$, sufficient for practical purposes to satisfy the condition, although it does not theoretically guarantee it. For each solution we verify whether it lies in the admissible region or not and the solution which does not will not be used. For the actual penalization settings, none of solutions obtained in our experiments lied outside the admissible region and, because of that, all restricted methods are comparable as they provide solutions under the same restrictions. For better detail, we measured the situation for a much smaller penalization of $c = 100$ and the smallest learning set size, where the danger of not satisfying the monotonicity conditions is the highest. Even in this case, only under 10% of initial solutions end outside the admissible region.

Binary question variables

In this section, we present results for the artificial model with binary question variables. The resulting values of the relative log-likelihood LR measured on the whole dataset for all set sizes are shown in Figure 3. Figure 4 then shows the KL divergence of the learned parameter distributions from the parameters of the generating distribution. In both figures, the horizontal axis has the logarithmic scale.

As we can see in Figure 3 all methods converge to the same log-likelihood value very quickly. Differences are mostly in the smaller set sizes of 10 and 100 observations. Unrestricted methods are clearly performing worse than the methods using the restrictions.

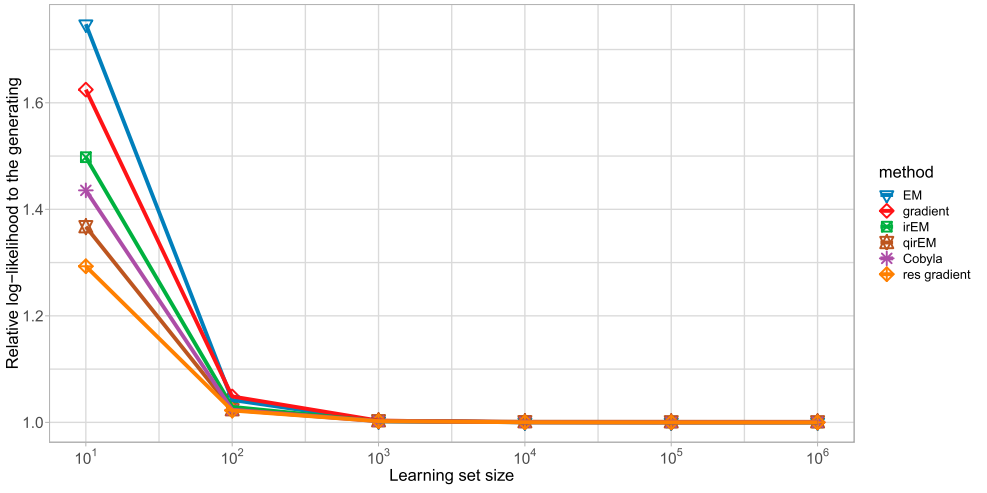


Figure 3. Artificial model, binary questions: The ratio between log-likelihoods of the fitted and the generating models.

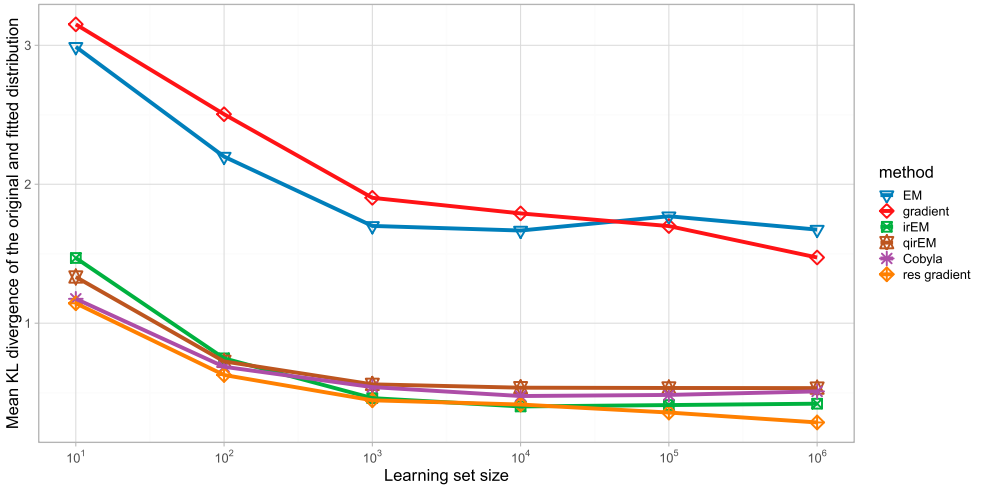


Figure 4. Artificial model, binary questions: The mean KL divergence of the fitted and generating probability distributions.

The isotonic regression and NLOPT methods provide similar results; and the methods of restricted gradient provide the best solutions for small sets. In the case of the KL divergence (in Figure 4), we can clearly see that monotonicity helps us obtain parameters which are closer to the real ones. In order to establish a sound ordering of the methods, we performed Wilcoxon's test. The null hypothesis was that one method is not giving better (lower) results. The *p*-values resulting from this test are presented in Table A1. We can see that, in most cases, the restricted gradient methods outperform the other methods at a significant level. The Cobyla and irEM methods are scoring very similarly against other methods but when pair-wise compared, the irEM is performing better.

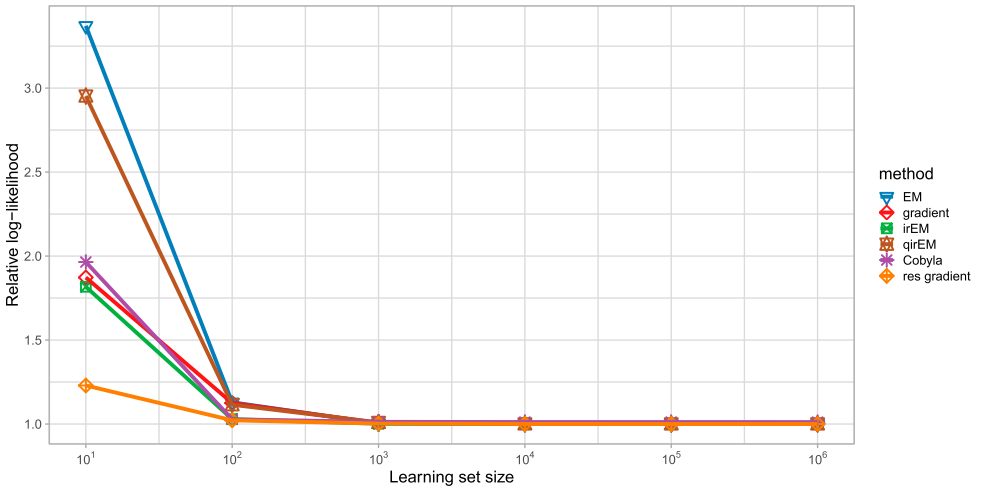


Figure 5. Artificial model, ternary questions: The ratio between log-likelihoods of the fitted and the generating models.

This model is small and all methods converge to a solution quite fast. Nevertheless, the EM and irEM methods are the fastest as they use the graph decomposition and update the CPTs separately. In the case of other methods, the structure remains complex (which is caused by unobserved parent variables) and such computations are more time-consuming for larger networks. The main problem is the increasing state space created by the state combinations of parents. As the number of parameters increases, the number of conditions increases as well, and computing the gradient also takes considerably more time. Especially in the case of NLOPT methods, this problem is significant.

Ternary question variables

The same testing scenario was used for ternary question variables. The results for relative log-likelihood are shown in Figure 5 and the divergence values of parameter distributions in Figure 6. These figures are constructed in the same way as those in the binary case. These results are very similar to the case with binary question variables. Wilcoxon's test results for this case are displayed in Table A2. They are almost identical to the previous case. The main difference is that the performance of the Cobyla method has significantly decreased. We can also observe that the order of methods is not exactly the same in both figures. The first figure shows the ability of methods to fit data. It is measured by the log-likelihood criterion. The second figure shows the distance of the fitted parameters to the parameters of the generating distribution. Models that have a high log-likelihood need not necessarily represent the best fit when it is measured by the distance of the parameters. Therefore we provide both views.

An interesting point to point out is that, unlike the other methods, the Cobyla one did not reach exactly the log-likelihood ratio 1 and the estimates of its parameters are leveled at an early stage. The reason for such behavior lies in the computational demands of this family of methods. The time which is sufficient for other methods is not sufficient for the Cobyla method to converge. We have also tested other possible methods from the NLOPT

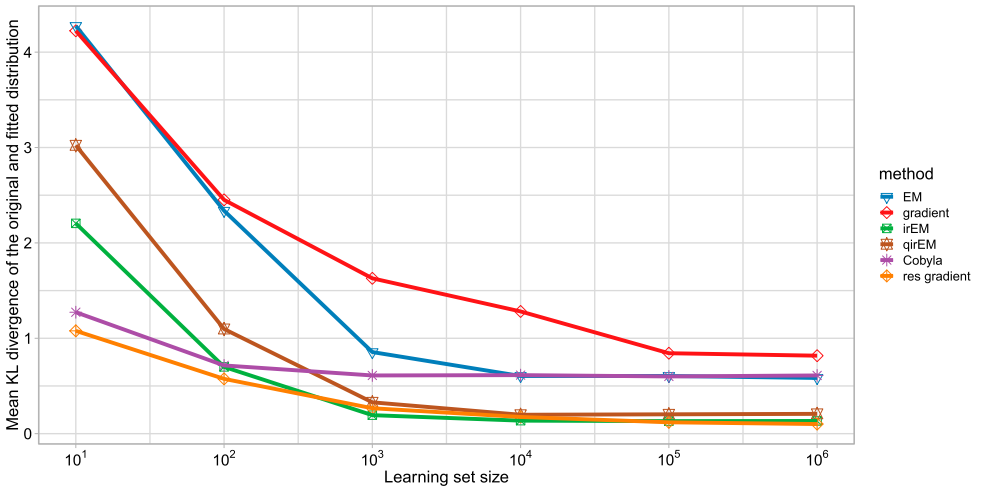


Figure 6. Artificial model, ternary questions: The mean KL divergence of the fitted and generating probability distributions.

family, including the global optimization method. The global optimization method had problems finding a solution even in the binary problem. Another NLOPT method, Sequential Least-Squares Quadratic Programming (SLSQP), which is a very fast local optimization method working well for the binary scenario, was not able to reach the solution either for this specific problem.

Example of isotonic regression EM behavior

We observed a problematic behavior of the irEM algorithm, which happens when the algorithm repeatedly leaves the space of admissible solutions during the EM step. We illustrate this behavior using a simple example. Figure 7 shows the log-likelihood during the fitting process. For the irEM algorithm, iterations are broken down into two consequent steps – the EM step and the isotonic regression step; the latter moves the parameters to the border of the admissible region. We can observe oscillations which are caused by leaving and re-entering the admissible region. This behavior creates an obstacle to finding the better solution in the similar way as local extremes do. The irEM algorithm fails to find a better solution which is reachable by the method from inside of the admissible space (starting parameters already monotonic). In Figure 8 we display the number of the violated monotonicity conditions. In fact, this behavior may also cause problems with the stopping criteria as the algorithm returns to the border possibly very close to the previous state after the ir step with a very similar log-likelihood value.

In Figure 7 we also present results of qirEM method. This method runs as the EM algorithm and performs the isotonic step after EM iterations. We can observe that the fitted log-likelihood is smaller than for the two other methods, but in the last step as the qirEM method satisfies the monotonicity conditions the log-likelihood rises above both concurrent methods. qirEM thus provides a valid solution but it is a heuristic which can potentially provide worse log-likelihood fits.

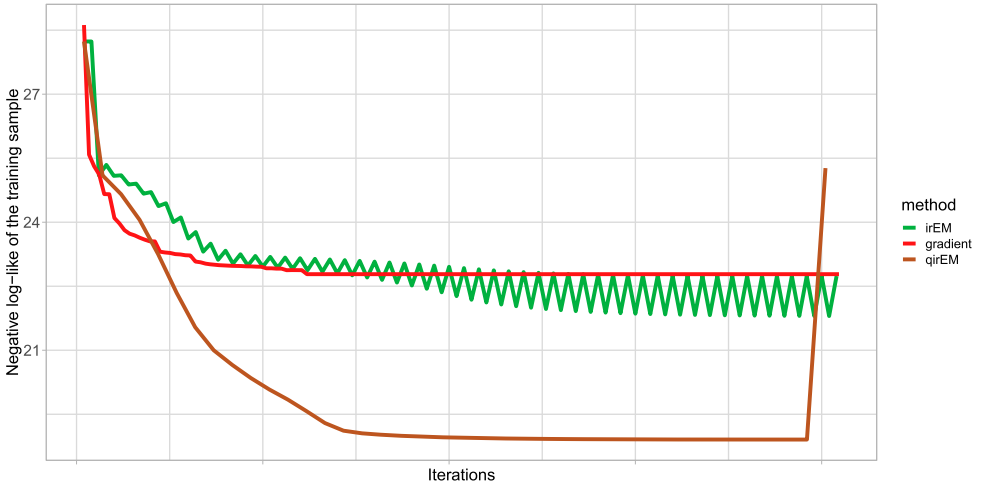


Figure 7. The evolution of the log-likelihood on the training sample during learning iterations of methods. The *x*-axis has not the same scale for all methods as the speed of convergence and the number of steps is not relevant in this case. For the irEM method we display both steps of the iteration in sequence (EM and ir).

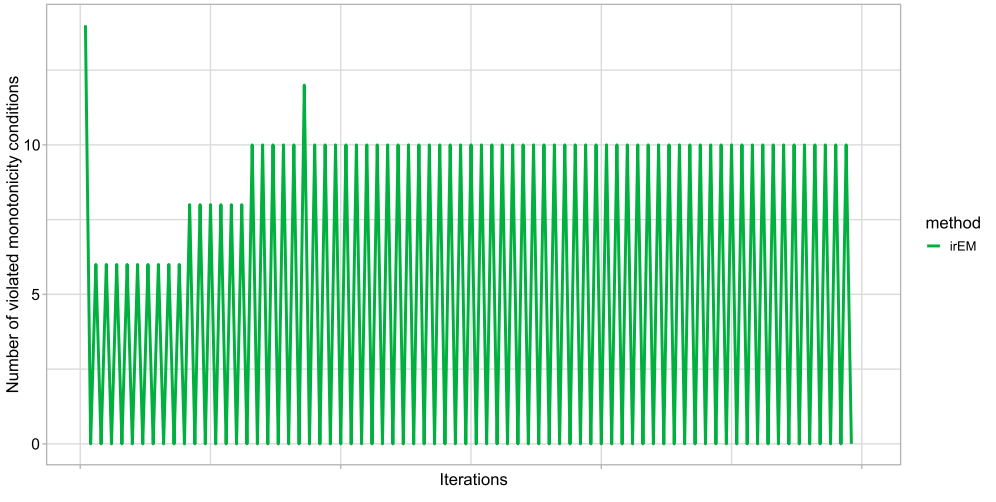


Figure 8. The evolution of the number of violated monotonicity constraints on the training sample for one example case – the binary artificial model. The irEM method displays both steps (EM and ir) in sequence.

4.2. CAT model

The structure of the second tested model is presented in Figure 2. Parent variables $S_1, \ldots, S_8$ have three states and each of them represents a particular student skill. Child nodes X_i are variables representing questions that have different numbers of states (based on the evaluation of the specific question). We learned this model from the data of the Czech high school final exam .[5] This dataset contains answers from over 20,000 students who took the test in the year 2015. We created the model structure based on our expert

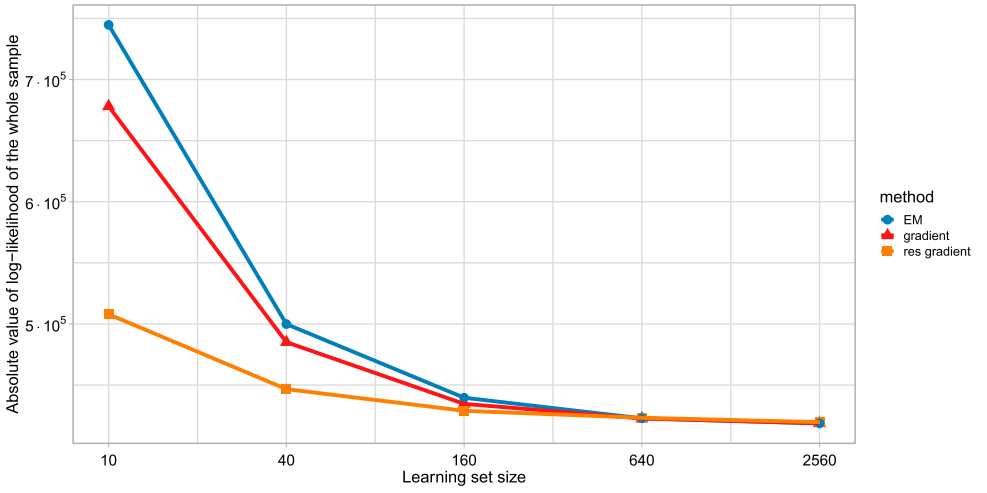


Figure 9. BN model for CAT empirical data: LLIK scored on the whole dataset for models trained with the EM and restricted gradient methods for different training set sizes. Notice the modified logarithmic scale on the x axis ($x = \log_4(x'/10)$).

analysis and assigned questions to relevant skills. We used random subsamples of the whole data sample with sizes of 10, 40, 160, 640, and 2560. We drew 10 random sets for each size. Models were initiated with 10 different random parameter vectors $\boldsymbol{\mu}_i$ and the corresponding $\boldsymbol{\theta}_i$.

This model was learned using our restricted gradient method and unrestricted gradient and EM methods for reference. In this case, we do not compare to the irEM method as we are not able to measure the divergence of the parameter distributions and the comparison would not be informative. The NLOPT family methods failed to obtain any solution in the given time (4 h).

We compute the log-likelihood of the learned models on the whole dataset. These values are then averaged similarly to the artificial model. The results are presented in Figure 9. In this case, we cannot compare the learned parameters with the real ones because the latter are unknown. In the figure, we can observe that, for empirical data, the restricted gradient methods provide better results for small datasets. The differences in the log-likelihood get lower for larger sets but, even in these cases, parameters of EM and unrestricted gradient learning are usually not monotonic. The parameter space is very large and these methods get easily trapped in a local optimum outside of the monotonicity region.

5. Conclusions

In this article, we present a new gradient-based method for learning parameters of Bayesian Networks under monotonicity restrictions. Our method is tested on two datasets. When considering the log-likelihood criterion, it is clearly visible in Figures 3, 5 and 9 that the new method provides better results than other methods for small training set sizes. When the size of the learning set grows, all methods are getting more accurate and fit the data better. The results obtained by all tested methods are very similar in terms of the log-likelihood criterion – except for the non-linear optimization approach, which in some cases, failed

to obtain any solution due to computational difficulties. For synthetic data, all methods converge to models with the same log-likelihood values, which are nearly identical with those of the log-likelihood of the generating model. In the case of empirical data, we can observe the same behavior. Again, for small training set sizes, the new gradient method is scoring better. All methods converge to identical log-likelihood values for large training data sets. Nevertheless, even for the large sets, parameters learned by a non-monotone method, such as EM or the unrestricted gradient, remain non-monotone. The parameter space is large and it is easy for these methods to get stuck in a local extreme with a not worse log-likelihood value but breaking the monotonicity conditions.

With the synthetic data generated from an artificial model, we are able to compare the fitted parameters with those of the generating model. These comparisons show that the newly proposed method is able to provide results which are closer to the original parameters in all cases. The only drawback of the new method is that it requires longer computational time than the irEM algorithm.

To summarize, we have shown that the learning methods can improve their behavior if they make use of valid monotonicity conditions. We have thus presented a new method that can be used to learn parameters of BNs under the monotonicity conditions. This method performs better in terms of the log-likelihood as well as of a distance from the original model parameters.

There are still open issues concerning the monotonicity conditions and learning parameters under them. One point to address is a generalization of our proposed algorithm to work on general BN structures rather than on bipartite graphs only. The potential application area of the general models is large. One example where we can use the monotonicity conditions regarding promotions planning is mentioned in the introduction of this paper. Another example is disease modeling where we could introduce the monotonicity to model increasing chances of a disease occurrence for higher levels of negative effects such as smoking. Hence it would be beneficial to further explore this research topic to provide larger possibilities to learn and use monotone models.

Notes

1. In our case, points are specifying the score obtained in the question i. The interpretation of points is very complex and has to be viewed as per question because we use the CAT framework. In this context, getting one point in one question is not the same as one point in another.
2. Note that for n_i this formula always holds since $\sum_{t=0}^{n_i} P(X_i = t|S_j = s_{j,k}, \boldsymbol{s}) = 1 \quad \forall i, \forall j, \forall k$.
3. As we use only reparameterized parameters in our gradient method, we provide only formulas with the reparametrization, i.e. the parameter vector $\boldsymbol{\mu}$ as was introduced in Section 2.2
4. The reason to include regular EM and unrestricted gradient methods is to further verify the benefit of using the monotonicity constraints. We want to provide a reader with a comparison also between the restricted and unrestricted cases.
5. The test assignment and its solution are accessible in the Czech language at: http://www.statnimaturita-matika.cz/wp-content/uploads/matematika-test-zadani-maturita-2015-jaro.pdf.

Disclosure statement

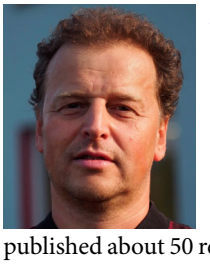
No potential conflict of interest was reported by the authors.

Funding

This work was supported by the Czech Science Foundation (Project No. 16-12010S and Project No. 19-04579S) and by the Grant Agency of the Czech Technical University in Prague [grant number SGS17/198/OHK4/3T/14].

Notes on contributors

Martin Plajner is a PhD student in Mathematical Engineering at the Czech Technical University in Prague, Faculty of Nuclear Sciences and Physical Engineering since 2013. He cooperates with the Institute of Information Theory and Automation of the Czech Academy of Sciences as a research PhD student under the supervision of Jiří Vomlel. His research interests are mainly centered on Computerized Adaptive Testing, Probabilistic Methods in Artificial Intelligence, and Bayesian Networks. He published research papers discussing these topics at various international conferences such as Conference on Probabilistic Graphical Models (PGM).



Jiří Vomlel obtained his PhD degree in Artificial Intelligence from the Czech Technical University in Prague. He spent three years at Aalborg University in Denmark where he worked as a research assistant in the research unit of Decision Support Systems. Since 2002, he has been a researcher in the Institute of Information Theory and Automation of the Czech Academy of Sciences. His researcher interests are Probabilistic Methods in Artificial Intelligence, Bayesian Networks, Computerized Adaptive Testing, Decision Theoretic Troubleshooting, and other applications of probabilistic graphical models. He has published about 50 research papers in scientific journals and peer-reviewed conference proceedings. He serves as an Area Editor of the *International Journal of Approximate Reasoning* and as regular program committee member of the Conference on Uncertainty in Artificial Intelligence (UAI), the International Joint Conference on Artificial Intelligence (IJCAI), and the International Conference on Probabilistic Graphical Models (PGM).

ORCID

Martin Plajner http://orcid.org/0000-0001-8388-1832
Jiří Vomlel http://orcid.org/0000-0001-5810-4038

References

Almond, R. G., and R. J. Mislevy. 1999. "Graphical Models and Computerized Adaptive Testing." *Applied Psychological Measurement* 23 (3): 223–237.

Altendorf, E. E., A. C. Restificar, and T. G. Dietterich. 2005. "Learning from Sparse Data by Exploiting Monotonicity Constraints." *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence (UAI2005).*

Druzdzel, J., and M. Henrion. 1993. "Efficient Reasoning in Qualitative Probabilistic Networks." In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, 548–553. AAAI Press.

Feelders, A. J. 2007. "A New Parameter Learning Method for Bayesian Networks with Qualitative Influences." In *UAI 2007, Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*, Vancouver, BC, Canada, July 19–22, 117–124.

Feelders, A. J., and L. C. van der Gaag. 2005. "Learning Bayesian Network Parameters with Prior Knowledge About Context-Specific Qualitative Influences." *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence (UAI2005).*

Hugin. 2014. "Explorer." Ver. 8.0. Comput. Software. http://www.hugin.com.

Johnson, S. G. 2018. "The NLopt Nonlinear-Optimization Package." Technical Report.

Kraft, D 1994. "Algorithm 733: TOMP–Fortran Modules for Optimal Control Calculations." *ACM Transactions on Mathematical Software* 20 (3): 262–281.
Masegosa, A. R., A. J. Feelders, and L. C. van der Gaag. 2016. "Learning from Incomplete Data in Bayesian Networks with Qualitative Influences." *International Journal of Approximate Reasoning* 69: 18–34.
Nielsen, T. D., and F. V. Jensen. 2007. *Bayesian Networks and Decision Graphs (Information Science and Statistics)*. New York: Springer-Verlag.
Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco: Morgan Kaufmann.
Plajner, M., and J. Vomlel. 2016a. "Probabilistic Models for Computerized Adaptive Testing: Experiments." Technical Report. ArXiv.
Plajner, M., and J. Vomlel. 2016b. "Student Skill Models in Adaptive Testing." In *Proceedings of the Eighth International Conference on Probabilistic Graphical Models*, 403–414. JMLR.org.
Plajner, M., and J. Vomlel. 2017. "Monotonicity in Bayesian Networks for Computerized Adaptive Testing." In *ECSQARU 2017*, edited by Alessandro Antonucci, Laurence Cholvy, and Odile Papini, 125–134. Cham: Springer.
Powell, M. J. D. 1994. "A Direct Search Optimization Method that Models the Objective and Constraint Functions by Linear Interpolation." In *Advances in Optimization and Numerical Analysis*, edited by S. Gomez and J.-P. Hennart, 51–67. Dordrecht: Kluwer Academic.
Rasch, G. 1960. *Studies in Mathematical Psychology: I. Probabilistic Models for Some Intelligence and Attainment Tests*. Copenhagen: Danmarks Paedagogiske Institut.
R Development Core Team. 2008. *R: A Language and Environment for Statistical Computing*. Vienna: R Foundation for Statistical Computing. ISBN 3-900051-07-0.
Restificar, A. C., and T. G. Dietterich. 2013. *Exploiting Monotonicity via Logistic Regression in Bayesian Network Learning*. Technical Report. Corvallis, OR : Oregon State University.
van der Gaag, L. C., H. L. Bodlaender, and A. J. Feelders. 2004. "Monotonicity in Bayesian Networks." In *20th Conference on Uncertainty in Artificial Intelligence (UAI '04)*, 569–576.
van der Gaag, L. C., and P. de Waal. 2006. "Multi-dimensional Bayesian Network Classifiers." *Proceedings of the Third European Workshop on Probabilistic Graphical Models (PGM06)*, Prague, 107–114.
van der Linden, W. J., and C. A. W. Glas. 2000. *Computerized Adaptive Testing: Theory and Practice*. Vol. 13. Dordrecht: Kluwer Academic.
Wellman, M. P 1990. "Fundamental Concepts of Qualitative Probabilistic Networks." *Artificial Intelligence* 44 (3): 257–303.

Appendix 1. Monotonicity restricted gradient

In the gradient descent optimization, we need partial derivatives to establish the gradient. The partial derivatives of $LL(\boldsymbol{\mu})$ with respect to $\mu_{i,\boldsymbol{s}^i}$ for $i \in \boldsymbol{N}, \boldsymbol{s}^i \in Val(\boldsymbol{S}^i)$ are

$$\frac{\partial LL(\boldsymbol{\mu})}{\partial \mu^t_{i,\boldsymbol{s}^i}} = \sum_{k \in \boldsymbol{D}} d_k \cdot \frac{I(t,i,\boldsymbol{s}^i,k) - \left(\left(\sum_{t'=0}^{n_i} \exp(\mu^{t'}_{i,\boldsymbol{s}^i})\right) - \exp(\mu^t_{i,\boldsymbol{s}^i})\right) \cdot p_{\tilde{\mu}}(\boldsymbol{\mu},\boldsymbol{s}^i) p_{\mu}(\boldsymbol{\mu},\boldsymbol{s}^i,k)}{\sum_{t'=0}^{n_i} \exp(\mu^{t'}_{i,\boldsymbol{s}^i}) \cdot \sum_{\boldsymbol{s} \in Val(\boldsymbol{S})} \left(p_{\tilde{\mu}}(\boldsymbol{\mu},\boldsymbol{s}) p_{\mu}(\boldsymbol{\mu},\boldsymbol{s},k)\right)},$$

where

$$I(t,i,\boldsymbol{s}^i,k) = \begin{cases} \exp(\mu^t_{i,\boldsymbol{s}^i}), & \text{if } t = k, \\ 0, & \text{otherwise,} \end{cases}$$

and with respect to $\tilde{\mu}_{j,l}$ for $j \in \boldsymbol{M}, l \in \{1, \ldots, m_j\}$ are

$$\frac{\partial LL(\boldsymbol{\mu})}{\partial \tilde{\mu}_{j,l}} = \sum_{k \in \boldsymbol{D}} d_k \cdot \frac{\sum_{\boldsymbol{s} \in \mathrm{Val}(\boldsymbol{S})}^{s_j = l} p_{\tilde{\mu}}(\boldsymbol{\mu}, \boldsymbol{s}) p_{\mu}(\boldsymbol{\mu}, \boldsymbol{s}, k)}{\sum_{\boldsymbol{s} \in \mathrm{Val}(\boldsymbol{S})} p_{\tilde{\mu}}(\boldsymbol{\mu}, \boldsymbol{s}) p_{\mu}(\boldsymbol{\mu}, \boldsymbol{s}, k)} - N \cdot \frac{\exp(\tilde{\mu}_{j,l})}{\sum_{l'=1}^{m_j} \exp(\tilde{\mu}_{k,l'})}.$$

The partial derivative of the penalization function $C(\boldsymbol{\mu}_{i,\boldsymbol{s}^i}, \boldsymbol{\mu}_{i,\boldsymbol{r}^i}, \hat{t}, c)$ is

$$\frac{\partial C(\boldsymbol{\mu}_{i,\boldsymbol{s}^i}, \boldsymbol{\mu}_{i,\boldsymbol{r}^i}, \hat{t}, c)}{\partial \mu^t_{i,\boldsymbol{s}^i}} = -p \cdot C_{p-1}(\boldsymbol{\mu}_{i,\boldsymbol{s}^i}, \boldsymbol{\mu}_{i,\boldsymbol{r}^i}, \hat{t}, c) \cdot \frac{g(\mu^t_{i,\boldsymbol{s}^i}, \hat{t}) \cdot \left(\sum_{t'=0}^{n_i} \exp(\mu^{t'}_{i,\boldsymbol{s}^i})\right) - \exp(\mu^t_{i,\boldsymbol{s}^i}) \cdot \sum_{t'=0}^{\hat{t}} \exp(\mu^{t'}_{i,\boldsymbol{s}^i})}{\left(\sum_{t'=0}^{n_i} \exp(\mu^{t'}_{i,\boldsymbol{s}^i})\right)^2},$$

where

$$g(\mu^t_{i,\boldsymbol{s}^i}, \hat{t}) = \begin{cases} 0, & \text{if } t > \hat{t}, \\ \exp(\mu^t_{i,\boldsymbol{s}^i}), & \text{if } t \leq \hat{t}, \end{cases}$$

and

$$\frac{\partial C(\boldsymbol{\mu}_{i,\boldsymbol{s}^i}, \boldsymbol{\mu}_{i,\boldsymbol{r}^i}, \hat{t}, c)}{\partial \mu^t_{i,\boldsymbol{r}^i}} = -\frac{\partial C(\boldsymbol{\mu}_{i,\boldsymbol{s}^i}, \boldsymbol{\mu}_{i,\boldsymbol{r}^i}, \hat{t}, c)}{\partial \mu^t_{i,\boldsymbol{s}^i}}.$$

The partial derivative of $LL'(\boldsymbol{\mu}, c)$ with respect to $\mu^t_{i,\boldsymbol{s}^i}$ is then

$$\frac{\partial LL'(\boldsymbol{\mu}, c)}{\partial \mu^t_{i,\boldsymbol{s}^i}} = \frac{\partial LL(\boldsymbol{\mu})}{\partial \mu^t_{i,\boldsymbol{s}^i}} - \sum_{\boldsymbol{s}^i \preceq_i \boldsymbol{r}^i} \sum_{\hat{t}=0}^{N^{\max}} \frac{\partial C\left(\boldsymbol{\mu}_{i,\boldsymbol{s}^i}, \boldsymbol{\mu}_{i,\boldsymbol{r}^i}, \hat{t}, c\right)}{\partial \mu^t_{i,\boldsymbol{s}^i}} - \sum_{\boldsymbol{r}^i \preceq_i \boldsymbol{s}^i} \sum_{\hat{t}=0}^{N^{\max}} \frac{\partial C\left(\boldsymbol{\mu}_{i,\boldsymbol{r}^i}, \boldsymbol{\mu}_{i,\boldsymbol{s}^i}, \hat{t}, c\right)}{\partial \mu^t_{i,\boldsymbol{s}^i}},$$

and the partial derivatives with respect to $\tilde{\mu}_{i,l}$ are not affected by the penalization as the parents do not appear in the penalization function.

$$\frac{\partial LL'(\boldsymbol{\mu})}{\partial \tilde{\mu}_{j,l}} = \frac{\partial LL(\boldsymbol{\mu})}{\partial \tilde{\mu}_{j,l}}.$$

Together

$$\frac{\partial LL'(\boldsymbol{\mu})}{\partial \tilde{\mu}_{j,l}} \quad \text{for } \{\tilde{\mu}_{j,l} | j \in \boldsymbol{M}, l \in \{1, \ldots, m_j\}\},$$

and

$$\frac{\partial LL'(\boldsymbol{\mu}, c)}{\partial \mu^t_{i,\boldsymbol{s}^i}} \quad \text{for } \{\mu^t_{i,\boldsymbol{s}^i} | i \in \boldsymbol{N}, t \in \{1, \ldots, N^{\max}\}\},$$

form the gradient $\nabla LL'(\boldsymbol{\mu}, c)$.

Appendix 2. Wilcoxon's tests

This appendix contains two tables, Table A1 and Table A2, with results of Wilcoxon's test for the KL divergences of generating and fitted probability distributions in artificial models with binary and ternary variables.

Table A1. Wilcoxon's test to compare results for artificial model with binary variables for different sizes of the learning sets. this test statistically verifies whether a method in the row statistically fits significantly better the generating parameters than another method in the column (H0: there is no shift in their distributions).

10	res gradient	irEM	qirEM	EM	gradient	Cobyla
res gradient	0.5005	**0.0000**	0.0099	**0.0000**	**0.0000**	0.1533
irEM	1.0000	0.5005	0.9986	**0.0000**	**0.0000**	1.0000
qirEM	0.9902	0.0014	0.5005	**0.0000**	**0.0000**	0.8996
EM	1.0000	1.0000	1.0000	0.5005	0.0708	1.0000
gradient	1.0000	1.0000	1.0000	0.9295	0.5005	1.0000
Cobyla	0.8473	**0.0000**	0.1009	**0.0000**	**0.0000**	0.5005
100						
res gradient	0.5005	**0.0002**	0.0055	**0.0000**	**0.0000**	0.0966
irEM	0.9998	0.5005	0.8708	**0.0000**	**0.0000**	0.9679
qirEM	0.9946	0.1297	0.5005	**0.0000**	**0.0000**	0.8624
EM	1.0000	1.0000	1.0000	0.5005	0.0025	1.0000
gradient	1.0000	1.0000	1.0000	0.9976	0.5005	1.0000
Cobyla	0.9038	0.0323	0.1382	**0.0000**	**0.0000**	0.5005
1000						
res gradient	0.5005	0.1313	**0.0000**	**0.0000**	**0.0000**	**0.0010**
irEM	0.8692	0.5005	**0.0000**	**0.0000**	**0.0000**	0.0099
qirEM	1.0000	1.0000	0.5005	**0.0000**	**0.0000**	0.9556
EM	1.0000	1.0000	1.0000	0.5005	0.0132	1.0000
gradient	1.0000	1.0000	1.0000	0.9869	0.5005	1.0000
Cobyla	0.9990	0.9902	0.0446	**0.0000**	**0.0000**	0.5005
1000						
res gradient	0.5005	0.8738	**0.0000**	**0.0000**	**0.0000**	0.0027
irEM	0.1267	0.5005	**0.0000**	**0.0000**	**0.0000**	0.0013
qirEM	1.0000	1.0000	0.5005	**0.0000**	**0.0000**	0.9951
EM	1.0000	1.0000	1.0000	0.5005	0.1048	1.0000
gradient	1.0000	1.0000	1.0000	0.8956	0.5005	1.0000
Cobyla	0.9973	0.9987	0.0050	**0.0000**	**0.0000**	0.5005
100,000						
res gradient	0.5005	0.2825	**0.0000**	**0.0000**	**0.0000**	**0.0000**
irEM	0.7183	0.5005	**0.0000**	**0.0000**	**0.0000**	0.0031
qirEM	1.0000	1.0000	0.5005	**0.0000**	**0.0000**	0.9984
EM	1.0000	1.0000	1.0000	0.5005	0.8101	1.0000
gradient	1.0000	1.0000	1.0000	0.1905	0.5005	1.0000
Cobyla	1.0000	0.9970	0.0017	**0.0000**	**0.0000**	0.5005
1,000,000						
res gradient	0.5151	0.1965	0.0026	**0.0000**	**0.0000**	0.0038
irEM	0.8237	0.5151	0.0827	**0.0000**	**0.0000**	0.1577
qirEM	0.9981	0.9284	0.5151	**0.0000**	**0.0000**	0.6981
EM	1.0000	1.0000	1.0000	0.5151	0.7255	1.0000
gradient	1.0000	1.0000	1.0000	0.3019	0.5177	0.9999
Cobyla	0.9972	0.8612	0.3304	**0.0000**	**0.0001**	0.5177

Table A2. Wilcoxon's test to compare results for artificial model with ternary variables for different sizes of the learning sets. this test statistically verifies whether a method in the row statistically fits significantly better the generating parameters than another method in the column (H0: there is no shift in their distributions).

10	res gradient	irEM	qirEM	EM	gradient	Cobyla
res gradient	0.5000	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
irEM	1.0000	0.5000	**0.0000**	**0.0000**	**0.0000**	1.0000
qirEM	1.0000	1.0000	0.5000	**0.0000**	**0.0000**	1.0000
EM	1.0000	1.0000	1.0000	0.5000	0.9995	1.0000
gradient	1.0000	1.0000	1.0000	**0.0005**	0.5000	1.0000
Cobyla	1.0000	**0.0000**	**0.0000**	**0.0000**	**0.0000**	0.5000
100						
res gradient	0.5000	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
irEM	1.0000	0.5000	**0.0000**	**0.0000**	**0.0000**	**0.0000**
qirEM	1.0000	1.0000	0.5000	**0.0000**	**0.0000**	1.0000
EM	1.0000	1.0000	1.0000	0.5000	**0.0000**	1.0000
gradient	1.0000	1.0000	1.0000	1.0000	0.5000	1.0000
Cobyla	1.0000	1.0000	**0.0000**	**0.0000**	**0.0000**	0.5000
1000						
res gradient	0.5001	0.8917	**0.0000**	**0.0000**	**0.0000**	**0.0000**
irEM	0.1084	0.5000	**0.0000**	**0.0000**	**0.0000**	**0.0000**
qirEM	1.0000	1.0000	0.5000	**0.0000**	**0.0000**	**0.0000**
EM	1.0000	1.0000	1.0000	0.5000	**0.0000**	1.0000
gradient	1.0000	1.0000	1.0000	1.0000	0.5000	1.0000
Cobyla	1.0000	1.0000	1.0000	**0.0000**	**0.0000**	0.5000
10,000						
res gradient	0.5001	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
irEM	1.0000	0.5000	**0.0000**	**0.0000**	**0.0000**	**0.0000**
qirEM	1.0000	1.0000	0.5000	**0.0000**	**0.0000**	**0.0000**
EM	1.0000	1.0000	1.0000	0.5000	**0.0000**	**0.0000**
gradient	1.0000	1.0000	1.0000	1.0000	0.5000	1.0000
Cobyla	1.0000	1.0000	1.0000	1.0000	**0.0000**	0.5000
100,000						
res gradient	0.5000	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
irEM	1.0000	0.5000	**0.0000**	**0.0000**	**0.0000**	**0.0000**
qirEM	1.0000	1.0000	0.5000	**0.0000**	**0.0000**	**0.0000**
EM	1.0000	1.0000	1.0000	0.5000	**0.0000**	**0.0000**
gradient	1.0000	1.0000	1.0000	1.0000	0.5000	1.0000
Cobyla	1.0000	1.0000	1.0000	1.0000	**0.0000**	0.5000
1,000,000						
res gradient	0.5005	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
irEM	1.0000	0.5005	**0.0000**	**0.0000**	**0.0000**	**0.0000**
qirEM	1.0000	1.0000	0.5005	**0.0000**	**0.0000**	**0.0000**
EM	1.0000	1.0000	1.0000	0.5005	0.0996	0.0260
gradient	1.0000	1.0000	1.0000	0.9009	0.5005	0.3323
Cobyla	1.0000	1.0000	1.0000	0.9742	0.6686	0.5005