Gradient Descent Parameter Learning of Bayesian Networks under Monotonicity Restrictions

Martin Plajner[1,2]
plajner@utia.cas.cz
Faculty of Nuclear Sciences and Physical Engineering[1],
Czech Technical University, Prague
Trojanova 13, Prague,
120 00, Czech Republic

Jiří Vomlel[2]
Institute of Information Theory and Automation[2],
Czech Academy of Sciences,
Pod vodárenskou věží 4, Prague 8, 182 08, Czech Republic

Abstract

Learning parameters of a probabilistic model is a necessary step in most machine learning modeling tasks. When the model is complex and data volume is small the learning process may fail to provide good results. In this paper we present a method to improve learning results for small data sets by using additional information about the modelled system. This additional information is represented by monotonicity conditions which are restrictions on parameters of the model. Monotonicity simplifies the learning process and also these conditions are often required by the user of the system to hold.

In this paper we present a generalization of the previously used algorithm for parameter learning of Bayesian Networks under monotonicity conditions. This generalization allows both parents and children in the network to have multiple states. The algorithm is described in detail as well as monotonicity conditions are.

The presented algorithm is tested on two different data sets. Models are trained on differently sized data subsamples with the proposed method and the general EM algorithm. Learned models are then compared by their ability to fit data. We present empirical results showing the benefit of monotonicity conditions. The difference is especially significant when working with small data samples. The proposed method outperforms the EM algorithm for small sets and provides comparable results for larger sets.

1 Introduction

In our research we address Computerized Adaptive Testing (CAT) [1, 13]. CAT is a concept of testing latent student abilities which allows us to create shorter tests, asking less questions in a shorter time while keeping the same level of information. This task is performed by asking the right questions for each individual student. Questions are selected based on a student model. In common practice experts often use Item Response Theory models [10] (IRT) 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 to model a student using Bayesian Networks (BNs) since they offer more options in the modelling process. It is for example possible to model more complex influences between skills and questions as BNs are not limited to connecting each skill with each question as well as we can introduce connections between skills themselves.

During our research we noticed that there are certain conditions which should be satisfied in this specific modelling task. We especially 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. It is addressed, selecting the most relevant to our topic, by [14, 3] and more recently by ,e.g., [11, 5]. 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 certain skills will have a higher probability of answering questions correctly.

Certain types of models include monotonicity naturally by the way they are constructed. In the case of general BNs this is not true. 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 first showed that monotonicity conditions are uself in the context of CAT [8]. Later we applied these conditions to Bayesian Network [9]. In this article we extend our earlier presented gradient descent optimum search method for BN parameter learning under monotonicity conditions. The last article covers only specific BNs. It works solely with binary children variables in the model (yes/no answers in terms of CAT). The extension we present in this article provides a tool to include monotonicity in BN models with multiple-state children nodes. Additionally, in this article we perform experiments on a new dataset. It is consists of data from the Czech high school state final exam. This data source contains a large volume of reliable data, and it is very useful for the empirical verification of our ideas.

We implemented the new method in R language and performed experimental verification of our assumptions. We used two data sets. The first one, a synthetic data set, is generated from artificial models satisfying monotonicity conditions. The second one, an empirical data set, is formed by data from the Czech high school final exam. Experiments were performed on these data sets also with the

ordinary EM learning without monotonicity restrictions in order to compare these two approaches.

The structure of this article is as follows. First, we establish our notation and describe monotonicity conditions in detail in Section 2. Next, we present the extended method in Section 3. In Section 4 of this paper, we take a closer look at the experimental setup and present results of our experiments. The last section contains an overview and a discussion of the obtained results.

2 BN Models and Monotonicity

2.1 Notation

In this article we use the new gradient descent method for BNs which are used to model students in the domain of CAT. Details about BNs can be found, for example, in [7, 6]. We restrict ourselves to the BNs that have two levels. In compliance with our previous articles, variables in the parent level are addressed as skill variables S. The children level contains questions 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 indexes 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 natural ordering specifying the number of points obtained in the $i-th$ question[1].
- 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 indexes of skill variables is $\boldsymbol{M} = \{1, \ldots, m\}$. Skill variables have a variable number of states, the total 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 only parent variables of the question X_i. Indexes 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 has CPT parameters for all questions $X_i, i \in \boldsymbol{N}, \boldsymbol{s}^i \in Val(\boldsymbol{S}^i)$ which define conditional probabilities as

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

and for all parent variables $S_j, j \in \boldsymbol{M}$ as

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

[1]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.

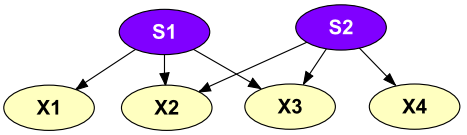

Figure 1: An artificial BN model

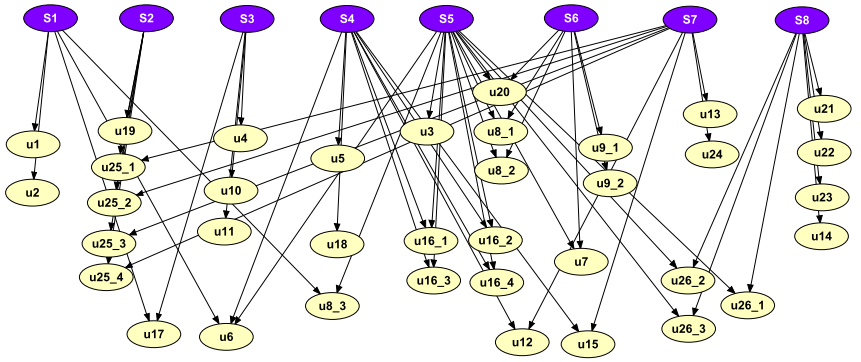

Figure 2: A BN model for CAT

From the definition above it follows that parameters are constrained to be between zero and one and to sum up to one. For question variable the condition is $\sum_{t=0}^{n_i} \theta^t_{i,\boldsymbol{s}^i} = 1,\ \forall i, \boldsymbol{s}^i$ and for parent variables it is $\sum_{s_j} \tilde{\theta}_{j,s_j} = 1,\ \forall j$. To remove this condition for the later use in the gradient method we reparametrize parameters

$$\begin{aligned} \theta^t_{i,\boldsymbol{s}^i} &= \frac{exp(\mu^t_{i,s_i})}{\sum_{t'=0}^{n_i} exp(\mu^{t'}_{i,s_i})} \\ \tilde{\theta}_{j,s_j} &= \frac{exp(\tilde{\mu}_{j,s_j})}{\sum_{s'_j=1}^{m_i} exp(\tilde{\mu}_{j,s'_j})} \quad . \end{aligned}$$

The set of all question parameters $\theta^t_{i,\boldsymbol{s}^i}$ and all skills parameters $\tilde{\theta}_{j,s_j}$ is $\boldsymbol{\theta}$ without the reparametrization and $\boldsymbol{\mu}$ with the reparametrization.

2.2 Monotonicity

The concept of monotonicity in BNs has been discussed in the literature since the last decade of the previous millennium [14, 3]. Later its benefits for BN parameter learning were addressed, for example, by [12, 2]. This topic is still active, e.g., [4, 11, 5].

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

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

A variable S_j has a monotone effect on its child X_i if for all $k, l \in \{1, \ldots, m_j\}, t' \in \{0, \cdots, n_i\}$:

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

and antitone effect:

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

where $\boldsymbol{s}$ is a configuration of 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 a monotone effect and by $\boldsymbol{S}^{i,-}$ the set of parents with an antitone effect.

The conditions above are defined for states of question variable X_i in the set $\{0, \cdots, (n_i - 1)\}$. Given the property of conditional probabilities, i.e.

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

it holds for the state n_i in the form for monotonic:

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

and for antitonic:

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

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

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

The monotonicity condition then requires that the probability of an incorrect answer is higher for a lower order parent configuration (chances of correct better answers increasing for higher ordered parents' states), i.e., for all $\boldsymbol{s}^i, \boldsymbol{r}^i \in Val(\boldsymbol{S^i}), k \in \{0, \ldots, (n_i - 1)\}$:

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

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

3 Parameter Gradient Search with Monotonicity

To learn the parameter vector $\boldsymbol{\mu}$ we have developed a method based on gradient descent optimization. We follow the work of [2] where authors use a gradient descent method with exterior penalties to learn parameters. The main difference is that we consider models with hidden variables. In this article we generalize the method from [9] to multistate question variables.

We denote by $\boldsymbol{D}$ the set of indexes of 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 model as described in Section 2 having unobserved parent variables and observed children variables. With sets $\boldsymbol{I}_t^k, t \in \{0, \ldots, N^{max}\}$ of indexes of questions answered with the point gain of t points, we define the following products based on observations in the k-th vector:

$$p^t(\boldsymbol{\mu}, \boldsymbol{s}, k) = \prod_{i \in \boldsymbol{I}_t^k} \frac{exp(\mu_{i,\boldsymbol{s}}^t)}{\sum_{t'=0}^{n_i} exp(\mu_{i,\boldsymbol{s}}^{t'})},\ t \in \{0, \cdots, N^{max}\}; \quad p_\mu(\boldsymbol{\mu}, \boldsymbol{s}) \quad = \quad \prod_{j=1}^{m} exp(\tilde{\mu}_{j,s_j}).$$

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

$$\begin{aligned} LL(\boldsymbol{\mu}) \quad &= \quad \sum_{k \in \boldsymbol{D}} d_k \cdot log\left(\sum_{\boldsymbol{s} \in Val(\boldsymbol{S})} \prod_{j=1}^{m} \frac{exp(\tilde{\mu}_{j,s_j})}{\sum_{s'_j=1}^{m_j} exp(\tilde{\mu}_{j,s'_j})} \cdot \prod_{t=0}^{N^{max}} p^t(\boldsymbol{\mu}, \boldsymbol{s}, k) \right) \\ &= \quad \sum_{k \in \boldsymbol{D}} d_k \cdot log\Big(\sum_{\boldsymbol{s} \in Val(\boldsymbol{S})} p_\mu(\boldsymbol{\mu}, \boldsymbol{s}) \prod_{t=0}^{N^{max}} p^t(\boldsymbol{\mu}, \boldsymbol{s}, k) \Big) - N \cdot \sum_{j=1}^{m} log \sum_{s'_j=1}^{m_j} exp(\tilde{\mu}_{j,s'_j}) \ . \end{aligned}$$

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

$$\frac{\delta LL(\boldsymbol{\mu})}{\delta \mu_{i,\boldsymbol{s^i}}^t} =$$

$$\sum_{k \in \boldsymbol{D}} d_k \cdot \quad \frac{I(t, i, \boldsymbol{s^i}, k) - (\sum_{t'=0}^{n_i} exp(\mu_{i,\boldsymbol{s^i}}^{t'}) - exp(\mu_{i,\boldsymbol{s^i}}^t)) \cdot p_\mu(\boldsymbol{\mu}, \boldsymbol{s}^i) \prod_{t=0}^{N^{max}} p^t(\boldsymbol{\mu}, \boldsymbol{s}, k)}{\sum_{t'=0}^{n_i} exp(\mu_{i,\boldsymbol{s^i}}^{t'}) \cdot \sum_{\boldsymbol{s} \in Val(\boldsymbol{S})} \left(p_\mu(\boldsymbol{\mu}, \boldsymbol{s}) \prod_{t=0}^{N^{max}} p^t(\boldsymbol{\mu}, \boldsymbol{s}, k) \right)},$$

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

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

$$\frac{\delta LL(\boldsymbol{\mu})}{\delta \tilde{\mu}_{i,l}} = \sum_{k \in \boldsymbol{D}} d_k \cdot \frac{\sum_{\boldsymbol{s} \in Val(\boldsymbol{S})}^{s_i=l} p_\mu(\boldsymbol{\mu}, \boldsymbol{s}) \prod_{t=0}^{N^{max}} p^t(\boldsymbol{\mu}, \boldsymbol{s}, k)}{\sum_{\boldsymbol{s} \in Val(\boldsymbol{S})} p_\mu(\boldsymbol{\mu}, \boldsymbol{s}) \prod_{t=0}^{N^{max}} p^t(\boldsymbol{\mu}, \boldsymbol{s}, k)} - \\ -N \cdot \frac{exp(\tilde{\mu}_{i,l})}{\sum_{l'=1}^{m_i} exp(\tilde{\mu}_{k,l'})} \ .$$

3.1 Monotonicity Restriction

To ensure monotonicity we use a penalty function which penalizes solutions that do not satisfy monotonicity conditions

$$C(\theta_{i,\boldsymbol{s}^i}, \theta_{i,\boldsymbol{r}^i}, t', c) = exp(c \cdot (\sum_{t=0}^{t'} \theta^t_{i,\boldsymbol{r}^i} - \sum_{t=0}^{t'} \theta^t_{i,\boldsymbol{s}^i}))$$

for the log likelihood:

$$LL'(\boldsymbol{\theta}, c) = LL(\boldsymbol{\theta}) - \sum_{i \in \boldsymbol{N}} \sum_{\boldsymbol{s}^i \preceq_i \boldsymbol{r}^i} \sum_{t'=0}^{N^{max}} C(\theta_{i,\boldsymbol{s}^i}, \theta_{i,\boldsymbol{r}^i}, t', c),$$

and in the case of reparametrized parameters:

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

where c is a constant determining the slope of the penalization function. The higher the value the more strict the penalization is. Theoretically, this condition does not ensure monotonicity but, practically, selecting high values of c results in monotonic estimates. If the monotonicity is not violated then the penalty value is close to zero. Otherwise, the penalty is raising exponentially fast. In our experiments we have used the value of $c = 200$ but any value higher than 100 provided almost identical results.

After adding the penalized part to the log likelihood, partial derivatives with respect to $\mu_{i,l}$ remain unchanged. Partial derivatives with respect to $\mu^t_{i,\boldsymbol{s}^i}$ change. The reparametrization causes the derivatives to become very complex. Due to limited space in this paper we do not include their full description here.

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 find the paramters of BN models.

4 Experiments

We designed tests to verify our assumptions. We want to show that if we learn parameters of BNs with little amount data it is beneficial to use monotonicity

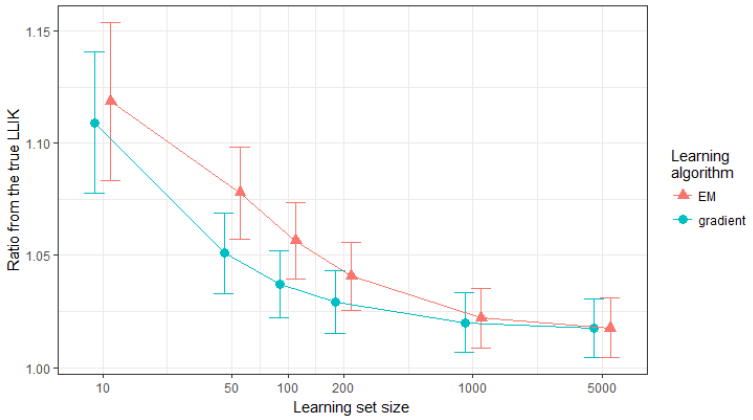

Figure 3: Artificial model: The ratio between the fitted and the real log likelihood (measured on the whole data set) obtained by models trained with EM and the restricted gradient methods for different training set sizes. Notice the logarithmic scale of the x axis. Curves are slightly misaligned in the direction of the x-axis to avoid overlapping.

constraints. We designed two experiments to test the method described above. The first one works with artificial (synthetic data); the other uses a real world empiric data sample.

Parameters are learned with our gradient method and the standard unrestricted EM algorithm. In both cases, we learn model parameters from subsets of data of different sizes. The quality of the parameter fit is measured by the log likelihood. The log likelihood is measured on the whole data set to provide results comparable between subsets of different sizes.

4.1 Artificial Model

The structure of the first model is shown in Figure 1. This model reflects the usual model structure used in CAT where there are two levels of variables, one level of questions, and one level of parents (skills). Parents S_1 and S_2 have 3 possible states and children X_1, X_2, X_3, X_4 also have three states. The model was set up with 10 different sets of parameters $\boldsymbol{\theta}^*_a$ satisfying the monotonicity conditions. Furthermore, every model produced 10 000 test cases.

To learn parameters of these models we drew random subsets of size d of 10, 50, 100, 200, 1 000, 5 000. Ten different sets for each size (indexed by b). Next, we created 10 initial starting points (indexed by c) for the model learning phase. The structure of both generating and learning models is the same and is shown in Figure 1. Starting parameter vectors $\boldsymbol{\theta}_b$ are randomized so that they satisfy the monotonicity conditions. Parameters of all parent variables are uniform. Starting

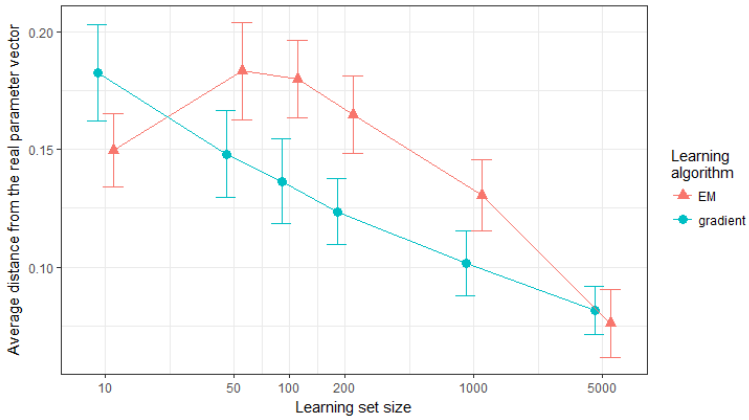

Figure 4: Artificial model: Mean parameter distance between real and fitted parameters in models trained with the EM and restricted gradient methods for different training set sizes. Notice the logarithmic scale of the x axis. Curves are slightly misaligned in the direction of the x-axis to avoid overlapping.

points are the same for both the EM and the gradient method alike. In this setup we have 10 different original models, 10 different observation subsets, and 10 different starting parameters, which gives 1 000 combinations for each set size. Each combination has a set of parameters $\boldsymbol{\theta}^d_{a,b,c}, a, b, c \in \{1, \ldots, 10\}$. We performed tests for all these combinations and the results are evaluated as follows.

We measure the log likelihood on the whole data set in order to keep results comparable. The resulting log likelihood after learning is compared with the log likelihood obtained with the real model and then averaged over all instances. This process gives us the average percentual difference between the original and fitted model. For the set size d:

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

Resulting valus for all set sizes are shown in Figure 3. In this artificial setup we are also able to measure the distance of learned parameters from the generating parameters. First we calculate an average error for each learned model:

$$e^d_{i,j} = \frac{|\boldsymbol{\theta}^*_a - \boldsymbol{\theta}^d_{a,b,c}|}{|\boldsymbol{\theta}|} \quad ,$$

where $||$ is the L1 norm. Next we average over all results in one set size d:

$$e^d = \frac{\sum_{a,b,c} e_{i,j}}{1000} \quad .$$

The summary of results is shown in Figure 4.

4.2 CAT Model

The second model is the model presented in Figure 2 and we use it for our CAT research. Parent variables $S_1, \ldots, S_8$ have 3 states and each one of them represents a particular student skill. Children nodes U_i are variables representing questions which have a various number of states (based on the evaluation of the specific question). This model was learned from data contained in the data sample collected from the Czech high school final exam[2]. The data set contains answers from over 20 000 students who took the test in the year 2015. We created the model structure based on our expert analysis and assigned skills to questions. To learn parameters we use random subsets of size of 10, 50, 100, and 500 cases of the whole sample. We drew 10 random sets for each size. Models were initiated with 10 different initial random starting parameters $\boldsymbol{\theta}_i$.

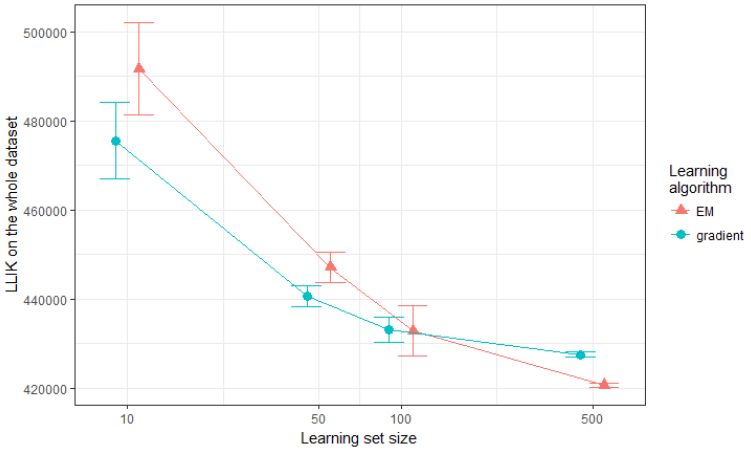

Figure 5: BN model for CAT empirical data: LLIK scored on the whole dataset for models trained with the EM and restricted gradient methods for different training set sizes. Notice the logarithmic scale of the x axis. Curves are slightly misaligned in the direction of the x-axis to avoid overlapping.

For the learned models we computed the log likelihood for the whole data set. These values are then averaged over all results of the same size $LL_A(k)$ similarly to the artificial model. Results are presented in Figure 5. In this case we cannot compare learned parameters because the real parameters are unknown.

[2]The test is accessible here (Czech language):`http://www.statnimaturita-matika.cz/wp-content/uploads/matematika-test-zadani-maturita-2015-jaro.pdf`

5 Conclusions

In this article we presented a new gradient based method for learning parameters of Bayesian Networks under monotonicity restrictions. The method was described and then tested on two data sets. In Figures 3 and 5 it is clearly visible that the newly proposed method provide better results than the general EM algorithm for small set sizes. When the size of learning set grows both method are getting more accurate and fitting data better. As we can see in results of the artificial model, both methods converge to the same point which is almost identical to the log likelihood of the model with real parameters. The speed of convergence is slower for the gradient method, nevertheless in the artificial case, it is not outperformed by the EM algorithm. In the case of empirical data, we can observe the same notion where for small set sizes the new gradient method is scoring better results. In this case EM is getting better log likelihood for larger data sets. This is caused by the fact that for these larger sets monotonicity restrictions start to make the learning process harder. For smaller sets they are showing the right path and guiding the learning process to a better solution. For larger sets they are restricting parameters and making the process harder. On the other hand, in case when we use the gradient method, we are working with learned model satisfying monotonicity conditions which may be desirable given its purpose.

This article shows that it is possible to benefit from monotonicity conditions. It presents the method to be used to learn parameter of BNs under these conditions. A possible extension of our work is to design a method which would use gradient descent optimization in a polytope defined by monotonicity conditions instead of using a penalty function. This approach has certain benefits as it ensures ending with strictly monotonic solution, on the other hand the current method allows small deviations from monotonicity if data strongly contradicts it.

Acknowledgements

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

References

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

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

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

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

[5] A. R. Masegosa, A. J. Feelders, and L. van der Gaag. Learning from incomplete data in Bayesian networks with qualitative influences. *International Journal of Approximate Reasoning*, 69:18–34, 2016.

[6] T. D. Nielsen and F. V. Jensen. *Bayesian Networks and Decision Graphs (Information Science and Statistics)*. Springer, 2007.

[7] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., dec 1988.

[8] M. Plajner and J. Vomlel. Student Skill Models in Adaptive Testing. In *Proceedings of the Eighth International Conference on Probabilistic Graphical Models*, pages 403–414. JMLR.org, 2016.

[9] M. Plajner and J. Vomlel. Monotonicity in Bayesian Networks for Computerized Adaptive Testing. In A. Antonucci, L. Cholvy, and O. Papini, editors, *ECSQARU 2017*, pages 125–134, Cham, 2017. Springer International Publishing.

[10] G. Rasch. *Studies in mathematical psychology: I. Probabilistic models for some intelligence and attainment tests.* Danmarks Paedagogiske Institut, 1960.

[11] A. C. Restificar and T. G. Dietterich. Exploiting monotonicity via logistic regression in Bayesian network learning. Technical report, Corvallis, OR : Oregon State University, 2013.

[12] L. van der Gaag, H. L. Bodlaender, and A. J. Feelders. Monotonicity in Bayesian networks. *20th Conference on Uncertainty in Artificial Intelligence (UAI '04)*, pages 569–576, 2004.

[13] W. J. van der Linden and C. A. W. Glas. *Computerized Adaptive Testing: Theory and Practice*, volume 13. Kluwer Academic Publishers, 2000.

[14] M. P. Wellman. Fundamental concepts of qualitative probabilistic networks. *Artificial Intelligence*, 44(3):257–303, 1990.