Monotonicity in Bayesian Networks for Computerized Adaptive Testing

Martin Plajner[1,2(✉)] and Jiří Vomlel[2]

[1] Faculty of Nuclear Sciences and Physical Engineering, Czech Technical University, Prague, Trojanova 13, 120 00 Prague, Czech Republic
[2] Institute of Information Theory and Automation, Czech Academy of Sciences, Pod Vodárenskou věží 4, 182 08 Prague 8, Czech Republic
{plajner,vomlel}@utia.cas.cz
http://staff.utia.cas.cz/plajner/
http://www.utia.cas.cz/vomlel/

Abstract. Artificial intelligence is present in many modern computer science applications. The question of effectively learning parameters of such models even with small data samples is still very active. It turns out that restricting conditional probabilities of a probabilistic model by monotonicity conditions might be useful in certain situations. Moreover, in some cases, the modeled reality requires these conditions to hold. In this article we focus on monotonicity conditions in Bayesian Network models. We present an algorithm for learning model parameters, which satisfy monotonicity conditions, based on gradient descent optimization. We test the proposed method on two data sets. One set is synthetic and the other is formed by real data collected for computerized adaptive testing. We compare obtained results with the isotonic regression EM method by Masegosa et al. which also learns BN model parameters satisfying monotonicity. A comparison is performed also with the standard unrestricted EM algorithm for BN learning. Obtained experimental results in our experiments clearly justify monotonicity restrictions. As a consequence of monotonicity requirements, resulting models better fit data.

Keywords: Computerized adaptive testing · Monotonicity · Isotonic regression EM · Gradient method · Parameters learning

1 Introduction

In our previous research Plajner and Vomlel (2015) we focused on Computerized Adaptive Testing (CAT) (Almond and Mislevy 1999; van der Linden and Glas 2000). We used artificial student models to select questions during the course of testing. We have shown that it is useful to include monotonicity conditions while learning parameters of these models (Plajner and Vomlel 2016b).

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.

A. Antonucci et al. (Eds.): ECSQARU 2017, LNAI 10369, pp. 125–134, 2017.
DOI: 10.1007/978-3-319-61581-3_12

Monotonicity conditions incorporate qualitative influences into a model. These influences restrict conditional probabilities in a specific way to avoid unwanted behavior. Some models we use for CAT include monotonicity naturally, but in this article we focus on a specific family of models, Bayesian Networks, which do not. Monotonicity in Bayesian Networks is discussed in literature for a long time. It is addressed, for example, by Wellman (1990), Druzdzel and Henrion (1993) and more recently by, e.g., Restificar and Dietterich (2013), Masegosa et al. (2016). Monotonicity restrictions are often motivated by reasonable demands from model users. In our case of CAT it means we want to make sure that students having certain skills will have a higher probability of answering questions depending on these skills correctly. Moreover, assuming monotonicity we can learn better models, especially when the data sample is small. In our work we have so far used monotonicity attained by logistic regression models of CPTs. This has proven useful but it is restrictive since it requires a prescribed CPT structure.

In this article we extends our results in the domain of Bayesian Networks. We present a gradient descent optimum search method for learning parameters of CPTs respecting monotonicity conditions. First, we establish our notation and monotonicity conditions in Sect. 2. Our method is derived in Sect. 3. We have implemented the method and performed tests. For testing we used two different data sets. First, we used a synthetic data set generated from a monotonic model (CPTs satisfying monotonicity) and second, we used real data set collected earlier. Experiments were performed on these data sets also with the isotonic regression EM (irem) method described by Masegosa et al. (2016) and the ordinary EM learning without monotonicity restrictions. In Sect. 4 of this paper we take a closer look at the experimental setup and present results of described tests. The last section brings an overview and a discussion of the obtained results.

2 BN Models and Monotonicity

2.1 Notation

In this article we use Bayesian Networks. Details about BNs can be found in, for example, Pearl (1988), Nielsen and Jensen (2007). We restrict ourselves to the following BN structure. Networks have two levels. In compliance with our previous articles, variables in the parent's level are addressed as skill variables S. The children level contains questions-answers variables X. Example network structures, which we also used for experiments, are shown in Figs. 1 and 2.

- We will use 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 are binary and they are observable.
- We will use 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 $\boldsymbol{M} = \{1, \ldots, m\}$.

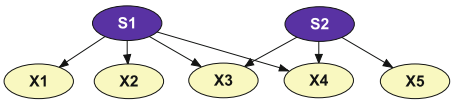

Fig. 1. Artificial model

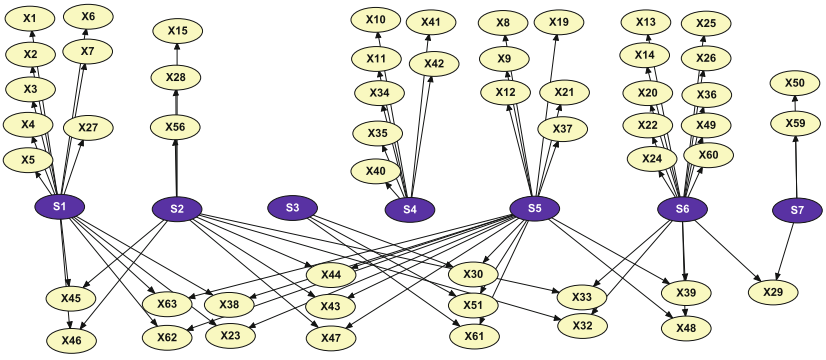

Fig. 2. CAT model network

Skill variables have variable number of states[1], 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 same as $\boldsymbol{S}$ but 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 all unobservable.

CPT parameters for a question variable X_i for all $i \in \boldsymbol{N}, \boldsymbol{s}^i \in Val(\boldsymbol{S}^i)$ are

$$\theta_{i,\boldsymbol{s}^i} = P(X_i = 0 | \boldsymbol{S}^i = \boldsymbol{s}^i),\ \boldsymbol{\theta}_i = (\theta_{i,\boldsymbol{s}^i})_{\boldsymbol{s}^i \in Val(\boldsymbol{S}^i)}.$$

We will also use $\theta_{i,\boldsymbol{s}} = \theta_{i,\boldsymbol{s}^i}$ with the whole parent set $\boldsymbol{S}$, where variables from $\boldsymbol{S} \backslash \boldsymbol{S}^i$ do not affect the value. Probabilities of a correct answer to a question X_i given state configuration $\boldsymbol{s}^i$ is $P(X = 1 | \boldsymbol{S}^i = \boldsymbol{s}^i) = 1 - \theta_{i,\boldsymbol{s}^i}$ (binary questions).

Parameters of parent variables for $j \in \boldsymbol{M}$ are

$$\rho_{j,s_j} = P(S_j = s_j),\ \boldsymbol{\rho}_j = (P(S_j = s_{j'}))\,,\ j' \in \{1, \ldots, m_j\}.$$

Parameter vector $\boldsymbol{\rho}_j$ is constrained by a condition $\sum_{s_j=1}^{m_j} \rho_{j,s_j} = 1$. To remove this condition we reparametrize this vector to

$$\rho_{j,s_j} = \frac{exp(\mu_{j,s_j})}{\sum_{s'_j=1}^{m_i} exp(\mu_{j,s'_j})}.$$

[1] In our experiments we use parents with 3 states, but the following theory applies to any number of states.

The whole vector of parameters is then

$$\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n, \boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_m), \text{ or } \boldsymbol{\mu} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_m),$$

where the meaning of $\boldsymbol{\mu_j}$ is the same as $\boldsymbol{\rho_j}$ but in this case vectors contain reparametrized variables. The transition from $\boldsymbol{\mu}$ to $\boldsymbol{\theta}$ is simply done with the reparametrization above and will be used without further notice. The total number of elements in the vector $\boldsymbol{\mu}$ and $\boldsymbol{\theta}$ is

$$l_{\boldsymbol{\mu}} = l_{\boldsymbol{\theta}} = \sum_{i \in \boldsymbol{N}} \prod_{j \in \boldsymbol{M}^i} m_j + \sum_{l \in \boldsymbol{M}} m_l.$$

2.2 Monotonicity

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

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

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

For questions we use natural ordering of its states ($0 \prec 1$).

A variable S_j has monotone, resp. antitone, effect on its child if for all $k, l \in \{1, \ldots, m_j\}$:

$$\begin{aligned} s_{j,k} \preceq s_{j,l} \Rightarrow P(X_i = 1 | S_j = s_{j,k}, \boldsymbol{s}) \;&\leq\; P(X_i = 1 | S_j = s_{j,l}, \boldsymbol{s}), \quad \text{resp.} \\ s_{j,k} \preceq s_{j,l} \Rightarrow P(X_i = 1 | S_j = s_{j,k}, \boldsymbol{s}) \;&>\; P(X_i = 1 | S_j = s_{j,l}, \boldsymbol{s}). \end{aligned}$$

where $\boldsymbol{s}$ is the configuration of other 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.

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

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

The monotonicity condition then requires that the question probability of correct answer is higher for a higher order parent configuration, i.e., for all $\boldsymbol{s}^i, \boldsymbol{r}^i \in Val(\boldsymbol{S^i})$:

$$\begin{aligned} &\boldsymbol{s}^i \preceq_i \boldsymbol{r}^i \Rightarrow P(X_i = 1 | \boldsymbol{S}^i = \boldsymbol{s}^i) \;\leq\; P(X_i = 1 | \boldsymbol{S}^i = \boldsymbol{r}^i), \\ &\boldsymbol{s}^i \preceq_i \boldsymbol{r}^i \Rightarrow P(Xi = 0 | \boldsymbol{S}^i = \boldsymbol{s}^i) \;\geq P(Xi = 0 | \boldsymbol{S}^i = \boldsymbol{r}^i) \;\Leftrightarrow\; \theta_{i,\boldsymbol{s}^i} \;\geq\; \theta_{i,\boldsymbol{r}^i}. \end{aligned}$$

In our experimental part we consider only isotone 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 parameter vector $\boldsymbol{\mu}$ we develop a method based on the gradient descent optimization. We follow the work of Altendorf et al. (2005) where they use a gradient descent method with exterior penalties to learn parameters. The main difference is that we consider models with hidden variables.

We denote by $\boldsymbol{D}$ the set of indexes of observations 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 structure as described in Sect. 2, i.e., unobserved parent variables and observed binary children variables. With sets $\boldsymbol{I}_0^k$ and $\boldsymbol{I}_1^k$ of indexes of incorrectly and correctly answered questions, we create following products based on observations in the k-th vector:

$$p_0^k(\boldsymbol{\mu}, \boldsymbol{s}, k) = \prod_{i \in \boldsymbol{I}_0^k} \theta_{i,\boldsymbol{s}}, \quad p_1^k(\boldsymbol{\mu}, \boldsymbol{s}, k) = \prod_{i \in \boldsymbol{I}_1^k} (1 - \theta_{i,\boldsymbol{s}}), \quad p_\mu(\boldsymbol{\mu}, \boldsymbol{s}) = \prod_{j=1}^{m} exp(\mu_{j,s_j}).$$

We work with the log likelihood:

$$\begin{aligned} LL(\boldsymbol{\mu}) &= \sum_{k \in \boldsymbol{D}} d_k \cdot log \left(\sum_{\boldsymbol{s} \in Val(\boldsymbol{S})} \prod_{j=1}^{m} \frac{exp(\mu_{j,s_j})}{\sum_{s'_j=1}^{m_j} exp(\mu_{j,s'_j})} \cdot p_0^k(\boldsymbol{\mu}, \boldsymbol{s}, k) \cdot p_1^k(\boldsymbol{\mu}, \boldsymbol{s}, k) \right) \\ &= \sum_{k \in \boldsymbol{D}} d_k \cdot log \Big(\sum_{\boldsymbol{s} \in Val(\boldsymbol{S})} p_\mu(\boldsymbol{\mu}, \boldsymbol{s}) \cdot p_0^k(\boldsymbol{\mu}, \boldsymbol{s}, k) \cdot p_1^k(\boldsymbol{\mu}, \boldsymbol{s}, k) \Big) \\ &\quad - N \cdot \sum_{j=1}^{m} log \sum_{s'_j=1}^{m_j} exp(\mu_{j,s'_j}). \end{aligned}$$

The partial derivatives of $LL(\mu)$ with respect to $\theta_{i,\boldsymbol{s}^i}$ for $i \in \boldsymbol{N}, \boldsymbol{s}^i \in Val(\boldsymbol{S}^i)$ are

$$\frac{\delta LL(\boldsymbol{\mu})}{\delta \theta_{i,\boldsymbol{s}^i}} = \sum_{k \in \boldsymbol{D}} d_k \cdot \frac{(-2x_i^k + 1) \cdot p_\mu(\boldsymbol{\mu}, \boldsymbol{s}^i) \cdot p_0^k(\boldsymbol{\mu}, \boldsymbol{s}^i, k) \cdot p_1^k(\boldsymbol{\mu}, \boldsymbol{s}^i, k)}{\theta_{i,\boldsymbol{s}^i} \cdot \sum_{\boldsymbol{s} \in Val(\boldsymbol{S})} p_\mu(\boldsymbol{\mu}, \boldsymbol{s}) \cdot p_0^k(\boldsymbol{\mu}, \boldsymbol{s}, k) \cdot p_1^k(\boldsymbol{\mu}, \boldsymbol{s}, k)}.$$

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

$$\begin{aligned} \frac{\delta LL(\boldsymbol{\mu})}{\delta \mu_{i,l}} &= \sum_{k \in \boldsymbol{D}} d_k \cdot \frac{\sum_{\boldsymbol{s} \in Val(\boldsymbol{S})}^{s_i=l} p_\mu(\boldsymbol{\mu}, \boldsymbol{s}) \cdot p_0^k(\boldsymbol{\mu}, \boldsymbol{s}, k) \cdot p_1^k(\boldsymbol{\mu}, \boldsymbol{s}, k)}{\sum_{\boldsymbol{s} \in Val(\boldsymbol{S})} p_\mu(\boldsymbol{\mu}, \boldsymbol{s}) \cdot p_0^k(\boldsymbol{\mu}, \boldsymbol{s}, k) \cdot p_1^k(\boldsymbol{\mu}, \boldsymbol{s}, k)} \\ &\quad - N \cdot \frac{exp(\mu_{i,l})}{\sum_{l'=1}^{m_i} exp(\mu_{k,l'})}. \end{aligned}$$

3.1 Monotonicity Restriction

To ensure monotonicity we use a penalty function

$$p(\theta_{i,\boldsymbol{s}^i}, \theta_{i,\boldsymbol{r}^i}) = exp(c \cdot (\theta_{i,\boldsymbol{r}^i} - \theta_{i,\boldsymbol{s}^i}))$$

for the log likelihood:

$$LL'(\boldsymbol{\mu}, c) = LL(\boldsymbol{\mu}) - \sum_{i \in \boldsymbol{N}} \sum_{\boldsymbol{s}^i \preceq_i \boldsymbol{r}^i} p(\theta_{i,\boldsymbol{s}^i}, \theta_{i,\boldsymbol{r}^i}),$$

where c is a constant determining the strength of the condition. Theoretically, this condition does not ensure monotonicity but, practically, selecting high values of c results in monotonic estimates. If the monotonicity is not violated, i.e. $\theta_{i,\boldsymbol{r}^i} < \theta_{i,\boldsymbol{s}^i}$ then the penalty value is close to zero. Otherwise, the penalty is raising exponentially fast with respect to $\theta_{i,\boldsymbol{r}^i} - \theta_{i,\boldsymbol{s}^i}$. In our experiments we have used the value of $c = 40$ but any value higher than 20 provided almost identical results.

Partial derivatives with respect to $\mu_{i,l}$ remain unchanged. Partial derivatives with respect to $\theta_{i,\boldsymbol{s}^i}$ are:

$$\frac{\delta LL'(\boldsymbol{\mu}, c)}{\delta \theta_{i,\boldsymbol{s}^i}} = \frac{\delta LL(\boldsymbol{\mu})}{\delta \theta_{i,\boldsymbol{s}^i}} + c \sum_{\boldsymbol{s}^i \preceq_i \boldsymbol{r}^i} p(\theta_{i,\boldsymbol{s}^i}, \theta_{i,\boldsymbol{r}^i}) - c \sum_{\boldsymbol{r}^i \preceq_i \boldsymbol{s}^i} p(\theta_{i,\boldsymbol{r}^i}, \theta_{i,\boldsymbol{s}^i})$$

Using the penalized log likelihood, $LL'(\boldsymbol{\mu}, c)$, and its gradient

$$\nabla(LL(\boldsymbol{\mu}, c)) = \Big(\frac{\delta LL'(\boldsymbol{\mu}, c)}{\delta \theta_{i,\boldsymbol{s}^i}}, \frac{\delta LL(\boldsymbol{\mu})}{\delta \mu_{j,l}}\Big),$$

for $i \in \boldsymbol{N}, \boldsymbol{s}^i \in Val(\boldsymbol{S}^i)$, $j \in \boldsymbol{M}, l \in \{1, \ldots, m_j\}$, we can apply the standard gradient method optimization to solve the problem. In order to ensure probability values of $\boldsymbol{\theta}_i, i \in \boldsymbol{N}$ it is necessary to use a bounded optimization method.

4 Experiments

For testing we use two different Bayesian Network models. The first one is an artificial model and we use simulated data. The second model is one of the models we used for computerized adaptive testing and we work with real data (for details please refer to Plajner and Vomlel (2016a)). In both cases we learn model parameters from data. Parameters are learned with our gradient method, isotonic regression EM[2] and the standard unrestricted EM algorithm. The learned model quality is measured by the log likelihood of the whole data sample including the training subset. This is done in order to provide results comparable between different training set sizes.

[2] We have implemented the irem algorithm based on the article (Masegosa et al. 2016). We extended the method to work with parents with more states than 2 (the article considers only binary variables). Questions (children) remain binary which makes the extension easy.

4.1 Artificial Model

The first model is displayed in Fig. 1. This model was created to provide simulated data for testing. The structure of the model is similar to models we use in CAT modeling with two levels of variables. Parents S_1 and S_2 have 3 possible states and children $X_1, \ldots, X_5$ are binary. We have instantiated the model with random parameters vector $\boldsymbol{\theta}^*$ satisfying monotonicity conditions. We drew a random sample of 100 000 cases from the model.

For parameters learning we use random subsets of size k of 10, 20, 50, 100, 1 000, 10 000, 50 000, and 100 000-(full data set) cases. For each size (except the last one) we use 10 different sets. Next, we prepared 15 initial parameter configurations for the fixed Bayesian Network structure (Fig. 1). These networks have starting parameters $\boldsymbol{\theta}_i$ generated at random, but in such a way, that they satisfy monotonicity conditions. The assumption of monotonicity is part of our domain expert knowledge. Therefore we can use it to speed up the process and avoid local optima. Parameters of parent variables are uniform and initial vectors are the same for each method. In our experiment we learn network parameters for each initial parameter setup for each set in a particular set size (giving a total of 150 learned networks for one set size). The learned parameter vectors are $\boldsymbol{\theta}_{i,j}$ for j-th subset of data.

The average log likelihood for the whole data sample

$$LL_A = \frac{\sum_{j=1}^{10} \sum_{i=1}^{15} LL(\boldsymbol{\theta}_{i,j})}{150}$$

is shown in Fig. 3 for each set size. In case of this model we are also able to measure the distance of learned parameters from the generating parameters in addition to the log likelihood. First we calculate an average error for each learned model:

$$e_{i,j} = \frac{|\boldsymbol{\theta}^* - \boldsymbol{\theta}_{i,j}|}{l_{\boldsymbol{\theta}}},$$

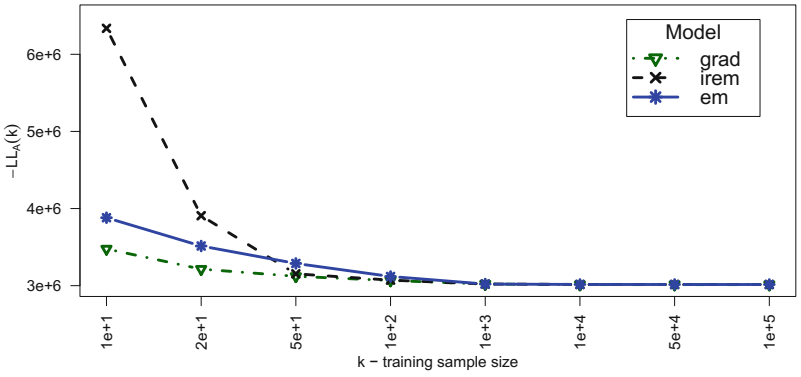

Fig. 3. Negative log likelihood for the whole sample and different training set sizes for the artificial model.

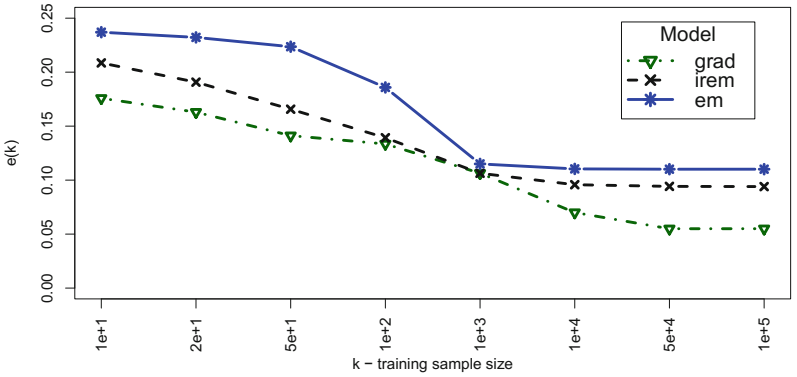

Fig. 4. Mean difference of parameters of learned and generating networks for different set sizes for the artificial model.

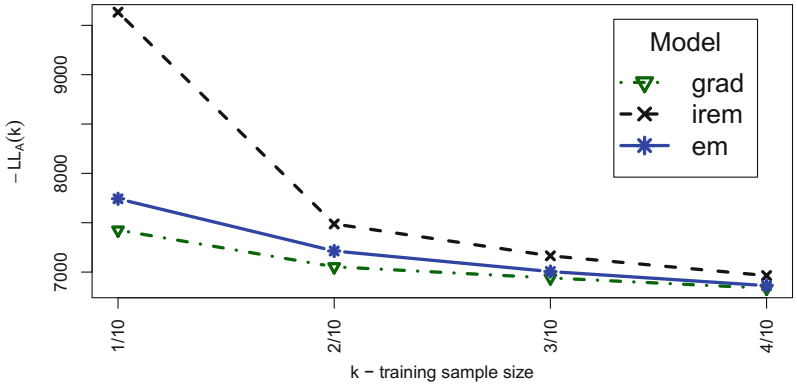

Fig. 5. Negative log likelihood for the whole sample and different training set sizes for the CAT model.

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

$$e = \frac{\sum_{j=1}^{10} \sum_{i=1}^{15} e_{i,j}}{150}.$$

Resulting values of e are displayed in Fig. 4 for each set size.

4.2 CAT Model

The second model is the model we used for CAT (Plajner and Vomlel 2016b). Its structure is displayed in Fig. 2. Parent variables $S_1, \ldots, S_7$ have 3 states and each one of them represents a particular student skill. Children nodes X_i are variables representing questions which are binary. Data associated with this model were collected from paper tests of mathematical skills of high school students.

In total the data sample has 281 cases. For more detailed overview of tests refer to Plajner and Vomlel (2016a). For learning we use random subsets of size of 1/10, 2/10, 3/10, and 4/10 of the whole sample. Similarly to the previous model, we drew 10 random sets for each size and initiated models by 15 different initial random monotonic starting parameters $\boldsymbol{\theta}_i$.

After learning we compute log likelihoods of the whole data set and we create averages for each set size $LL_A(k)$ as with the previous model. Resulting values are in Fig. 5. In this case we cannot compare learned parameters because the real parameters with real are unknown.

5 Conclusions

In this article we have presented a gradient based method for learning parameters of Bayesian Network under monotonicity restrictions. The method was described and then tested on two data sets. In Figs. 3 and 5 it is clearly visible that this method achieves the best results from three tested methods (especially for small training samples). The irem method has problems with small training samples and the log likehood in those cases is low. This is a consequence of the fact that it moves to monotonic solution from a poor EM estimate and in these cases ensuring monotonicity implies log likelihood degradation. We can also observe that for the training sets larger than 1000 data vectors the EM algorithm stabilizes in its parameter estimations. It means that at about $k = 1000$ the EM algorithm found the best model it can and increasing training size does not improve the result. Nevertheless, as we can observe in Fig. 4 parameters of learned networks are always closer to the generating parameters while considering monotonicity for both the irem and the gradient methods than for the standard EM.

These results verify usefulness of monotonicity for learning Bayesian Networks. A possible extension is to enlarge the theory of gradient based method to work with more general network structures.

References

Almond, R.G., Mislevy, R.J.: Graphical models and computerized adaptive testing. Appl. Psychol. Meas. **23**(3), 223–237 (1999)

Altendorf, E.E., Restificar, A.C., Dietterich, T.G.: Learning from sparse data by exploiting monotonicity constraints. In: Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence (UAI 2005) (2005)

Druzdzel, J., Henrion, M.: Efficient reasoning in qualitative probabilistic networks. In: Proceedings of the Eleventh National Conference on Artificial Intelligence, pp. 548–553. AAAI Press (1993)

Feelders, A.J., van der Gaag, L.: Learning Bayesian network parameters with prior knowledge about context-specific qualitative influences. In: Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence (UAI 2005) (2005)

Masegosa, A.R., Feelders, A.J., van der Gaag, L.: Learning from incomplete data in Bayesian networks with qualitative influences. Int. J. Approx. Reason. **69**, 18–34 (2016)

Nielsen, T.D., Jensen, F.V.: Bayesian Networks and Decision Graphs. Information Science and Statistics. Springer, New York (2007)

Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann Publishers Inc., San Francisco (1988)

Plajner, M., Vomlel, J.: Bayesian network models for adaptive testing. In: Proceedings of the Twelfth UAI Bayesian Modeling Applications Workshop, pp. 24–33. CEUR-WS.org, Amsterdam (2015)

Plajner, M., Vomlel, J.: Probabilistic models for computerized adaptive testing: experiments. Technical report, arXiv:1601.07929 (2016a)

Plajner, M., Vomlel, J.: Student skill models in adaptive testing. In: Proceedings of the Eighth International Conference on Probabilistic Graphical Models, pp. 403–414. JMLR.org (2016b)

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

van der Gaag, L., Bodlaender, H.L., Feelders, A.J.: Monotonicity in Bayesian networks. In: 20th Conference on Uncertainty in Artificial Intelligence (UAI 2004), pp. 569–576 (2004)

van der Linden, W.J., Glas, C.A.W.: Computerized Adaptive Testing: Theory and Practice, vol. 13. Kluwer Academic Publishers, Dordrecht (2000)

Wellman, M.P.: Fundamental concepts of qualitative probabilistic networks. Artif. Intell. **44**(3), 257–303 (1990)