Pattern Recognition 42 (2009) 3046 – 3056

Contents lists available at ScienceDirect

Pattern Recognition

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

Learning Bayesian network parameters under incomplete data with domain knowledge

Wenhui Liao[a,*], Qiang Ji[b]

[a]Thomson Reuters, Eagan, MN 55123, USA
[b]ECSE, Rensselaer Polytechnic Institute, Troy, NY 12180, USA

ARTICLE INFO

Article history:
Received 24 April 2008
Received in revised form 4 February 2009
Accepted 7 April 2009

Keywords:
Bayesian network parameter learning
Missing data
EM algorithm
Facial action unit (AU) recognition

ABSTRACT

Bayesian networks (BNs) have gained increasing attention in recent years. One key issue in Bayesian networks is parameter learning. When training data is incomplete or sparse or when multiple hidden nodes exist, learning parameters in Bayesian networks becomes extremely difficult. Under these circumstances, the learning algorithms are required to operate in a high-dimensional search space and they could easily get trapped among copious local maxima. This paper presents a learning algorithm to incorporate domain knowledge into the learning to regularize the otherwise ill-posed problem, to limit the search space, and to avoid local optima. Unlike the conventional approaches that typically exploit the quantitative domain knowledge such as prior probability distribution, our method systematically incorporates qualitative constraints on some of the parameters into the learning process. Specifically, the problem is formulated as a constrained optimization problem, where an objective function is defined as a combination of the likelihood function and penalty functions constructed from the qualitative domain knowledge. Then, a gradient-descent procedure is systematically integrated with the E-step and M-step of the EM algorithm, to estimate the parameters iteratively until it converges. The experiments with both synthetic data and real data for facial action recognition show our algorithm improves the accuracy of the learned BN parameters significantly over the conventional EM algorithm.

1. Introduction

In recent years, Bayesian networks (BNs) have been increasingly used in a wide range of applications including computer vision [1], bioinformatics [2], information retrieval [3], data fusion [4], decision support systems and others. A BN is a directed acyclic graph (DAG) that represents a joint probability distribution among a set of variables, where the nodes denote random variables and the links denote the conditional dependencies among variables. The advantages of BNs can be summarized as their semantic clarity and understandability by humans, the ease of acquisition and incorporation of prior knowledge, the possibility of causal interpretation of learned models, and the automatic handling of noisy and missing data [5].

In spite of these claims, people often face the problem of learning BNs from training data in order to apply BNs to real-world applications. Typically, there are two categories in learning BNs, one is to learn BN parameters when a BN structure is known, and another is to learn both BN structures and parameters. In this paper, we focus on BN parameter learning by assuming the BN structure is already known. If the training data is complete, learning BN parameters is not difficult, however, in real world, training data can be incomplete for various reasons. For example, in a BN modeling video surveillance, the training data may be incomplete because of security issue; in a BN modeling customer behaviors, the training data may be incomplete because of privacy issue. Sometimes, the training data may be complete but sparse, because some events rarely happen, or the data for these events are difficult to obtain.

In general, training data can be missed in three ways: *missing at random (MAR)*, *missing completely at random (MCAR)*, and *not missing at random* (NMAR). MAR means the probability of missing data on any variable is not related to its particular value, but could be related to other variables. MCAR means the missing value of a variable depends neither on the variable itself nor on the values of other variables in the BN. For example, some hidden (latent) nodes never have data. NMAR means data missing is not at random but depends on the values of the variables.

The majority of the current learning algorithms assume the MAR property holds for all the incomplete training samples since learning is easier for MAR than NMAR and MCAR. The classical approaches

* Corresponding author.
E-mail addresses: wenhui.liao@thomsonreuters.com (W. Liao), qji@ecse.rpi.edu (Q. Ji).

0031-3203/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2009.04.006

include the Expectation Maximization (EM) algorithm [6] and Gibbs sampling [7]. Other methods are proposed to overcome the disadvantages of EM and Gibbs sampling. For example, methods are proposed to learn the parameters when data are not missing at random, such as the AI&M procedure [8], the RBE algorithm [9], and the maximum entropy method [10,11]; some methods are proposed to escape local maxima under the assumption of MAR, such as the information-bottleneck EM (IB-EM) algorithm [12], data perturbation method [13], etc.; other methods are proposed to speed up the learning procedure, such as generalized conjugate gradient algorithm [14], online updating rules [15], and others.

When data are missing completely at random, in other words, when several hidden nodes exist, those methods could fail, where the learned parameters may be quite different from the true parameters. In fact, since there are no data for hidden nodes, learning parameters becomes an ill-posed problem. Thus, prior data on domain knowledge are needed to regularize the learning problem. In most domains, at least some information, either from literature or from domain experts, is available about the model to be constructed. However, many forms of prior knowledge that an expert might have are difficult to be directly used by existing machine learning algorithms. Therefore, it is important to formalize the knowledge systematically and incorporate it into the learning. Such domain knowledge can help regularize the otherwise ill-posed learning problem, reduce the search space significantly, and help escape local maxima.

This motivates us to propose a Bayesian network learning algorithm for the case when multiple hidden nodes exist by systematically combining domain knowledge during learning. Instead of using quantitative domain knowledge, which is often hard to obtain, we propose to exploit qualitative domain knowledge. Qualitative domain knowledge impose approximated constraints on some parameters or on the relationships among some parameters. These kind of qualitative knowledge are often readily available. Specifically, two qualitative constraints are considered, the range of parameters, and the relative relationships between different parameters. Instead of using the likelihood function as the objective to maximize during learning, we define the objective function as a combination of the likelihood function and the penalty functions constructed from the domain knowledge. Then, a gradient-descent procedure is systematically integrated with the Expectation-step (E-step) and Maximization-step (M-step) of the EM algorithm, to estimate the parameters iteratively until it converges. The experiments show the proposed algorithm significantly improves the accuracy of the learned BN parameters over the conventional EM method.

2. Related work

During the past several years, many methods have been proposed to learn BN parameters when data are missing. Two standard learning algorithms are Gibbs sampling [7] and EM [6]. Gibbs sampling by Geman and Geman [7] is the basic tool of simulation and can be applied to virtually any graphical model whether the arcs are directed or not, and whether the variables are continuous or discrete [16]. It completes the samples by inferring the missing data from the available information and then learns from the completed database (imputation strategy). Unfortunately, Gibbs sampling method suffers from convergence problems arising from correlations between successive samples [10]. In addition, it is not effective when data are missing in complete random (e.g. the case of the hidden nodes).

The EM algorithm can be regarded as a deterministic version of Gibbs sampling used to search for the Maximum Likelihood (ML) or Maximum a Posteriori (MAP) estimate for model parameters [16,6]. However, when there are multiple hidden variables or a large amount of missing data, EM gets easily trapped in a local maximum. "With data missing massively and systematically, the likelihood function has a number of local maxima and straight maximum likelihood gives results with unsuitably extreme probabilities" [17]. In addition, EM algorithms are sensitive to the initial starting points. If the initial starting points are far away from the optimal solution, the learned parameters are not reliable.

Different methods are proposed to help avoid local maxima. Elidan and Friedman [12] propose an information-bottleneck EM (IB-EM) algorithm to learn the parameters of BNs with hidden nodes. It treats the learning problem as a tradeoff between two information-theoretic objectives, where the first one is to make the hidden nodes uninformative about the identity of specific instances, and the second one is to make the hidden variables informative about the observed attributes. However, although IB-EM has a better performance than the standard EM for some simple BNs, it is actually worse than EM for the complex hierarchical models as shown in [12]. To escape local maxima in learning, Elida et al. [13] propose a solution by perturbing training data. Two basic techniques are used to perturb the weights of the training data: (1) random reweighing, which randomly samples weight profiles on the training data, and (2) adversarial reweighing, which updates the weight profiles to explicitly punish the current hypothesis, with the intent of moving the search quickly to a nearby basin of attraction. Although it usually achieves better solutions than EM, it is still a heuristic method and not necessarily able to escape local maxima. And also, it is much slower than the standard EM algorithm.

The previous methods emphasize improving the machine learning techniques, instead of using domain knowledge to help learning. Since there are no data available for hidden nodes, it is important to incorporate any available information about these nodes into learning. The methods for constraining the parameters for a BN include Dirichlet priors, parameter sharing, and qualitative constraints. According to [18], there are several problems using Dirichlet priors. First, it is impossible to represent even the simple equality constraints on the parameters. Second, it is often beyond expert's capability to specify a full Dirichlet prior over the parameters of a Bayesian network. Parameter sharing, on the other hand, allows parameters of different models to share the same values, i.e., it allows to impose equality constraints. Parameter sharing methods, however, do not capture more complicated constraints among parameters such as inequality constraints among the parameters. In addition, both Dirichlet priors and parameter sharing methods are restricted to sharing parameters at the level of sharing a whole CPT or CPTs, instead of at the level of granularity of individual parameters. To overcome these limitations, others [19–22,18] propose to explicitly exploit qualitative relationships among parameters and systematically incorporate them into the parameter estimation process.

Druzdel et al. [19] give formal definitions of several types of qualitative relationships that can hold between nodes in a BN to help specify CPTs of BNs, including probability intervals, qualitative influences, and qualitative synergies. They express these available information in a canonical form consisting of (in)equalities expressing constraints on the hyperspace of possible joint probability distributions, and then use this canonical form to derive upper and lower bounds on any probability of interest. However, the upper and lower bounds cannot give sufficient insight into how likely a value from the interval is to be the actual probability.

Wittig and Jameson [20] present a method for integrating formal statements of qualitative constraints into two learning algorithms, APN [23,24] and EM. Two types of qualitative influences [19] are considered as constraints for parameters during learning in this method: (1) a positive influence holds between two variables (X_1, X_2) if for any given value of X_2, an increase in the value of X_1 will not decrease the probability that the value of X_2 is equal to or greater than that given value; and (2) a negative influence can be defined analogously.

This paper shows that the accuracy of the learned BNs is superior to that of the corresponding BNs learned without the constraints.

Even when data are complete, domain knowledge can help learning significantly, in particular when insufficient data are available. Altendorf et al. [21] show how to interpret knowledge of qualitative influences, and in particular of monotonicities, as constraints on probability distribution, and to incorporate this knowledge into BN learning algorithms. It assumes that the values of the variables can be totally ordered. It focuses on learning from complete but sparse data. It shows the additional constraints provided by qualitative monotonicities can improve the performance of BN classifiers, particularly on extremely small training sets. The assumption that the values of the variables can be totally ordered is, however, too restrictive.

Feelders and Gaag [22,25] present a method for learning BN parameters with prior knowledge about the signs of influences between variables. The various signs are translated into order constraints on the network parameters and isotonic regression is then used to compute order-constrained estimates from the available data. They also focus only on learning from complete but sparse data. In addition, they assume all variables are binary.

The constraints analyzed in these papers [20–22] are somewhat restrictive, in the sense that each constraint must involve all parameters in a conditional probability table (CPT). Niculescu [18] considers a different type of domain knowledge for constraining parameter estimate when data are complete but limited. Two types of constraints are used: one is that the sum of several parameters within one conditional probability distribution is bounded by the sum of other parameters in the same distribution; another is an upper bound on the sum of several parameters in one conditional probability distribution. Both the constraints have to be twice differentiable, with continuous second derivatives. They formalize the learning as a constraint optimization problem and derive closed form maximum likelihood parameter estimators.

Our learning method also exploits domain knowledge to help parameter learning, but especially for BNs with hidden nodes or a large amount of missing data. Different from other learning methods, the domain knowledge we used has the following features: (1) it does not need to involve all the parameters in a conditional probability table; (2) it can deal with relationships between different parameters; (3) it is associated with confidence levels to reflect the confidence of the domain experts; (4) it is easy to assess and define; and (5) new format of domain knowledge can be easily incorporated into the algorithm. Our algorithm systematically incorporates the domain knowledge and is capable of learning parameters successfully even when a large percent of nodes are hidden in a Bayesian network.

3. Learning Bayesian network parameters

3.1. The basic theory

Let G be a BN with nodes $X_1,\ldots,X_n$. If there is a directed arc from X_i to X_j, X_i is called a parent of X_j, $pa(X_j)$. Given its parent nodes, a node is conditionally independent from all the other nodes. Thus the joint distribution of all the nodes can be written as

$$p(X_1,\ldots,X_n)=\prod_{i=1}^{n} p(X_i|pa(X_i)) \tag{1}$$

Each node is associated with several parameters to describe the conditional probability distribution of the random variable given its parents. We use θ to denote the entire vector of parameter value θ_{ijk}, $\theta_{ijk}=p(x_i^k|pa_i^j)$, where i ($i=1,\ldots,n$) ranges over all the variables in the BN, j ($j=1,\ldots,q_i$) ranges over all the possible parent configurations of X_i, and k ($k=1,\ldots,r_i$) ranges over all the possible states of X_i. Therefore, x_i^k represents the kth state of variable X_i, and pa_i^j is the jth configuration of the parent nodes of X_i.

Given the data set $D=\{D_1,\ldots,D_N\}$, where $D_l=\{x_1[l],\ldots,x_n[l]\}$ that consists of instances of the BN nodes, the goal of parameter learning is to find the most probable value $\hat{\theta}$ for θ that can best explain the data set D, which is usually quantified by the log-likelihood function, $log(p(D|\theta))$, denoted as $L_D(\theta)$. If D is complete, based on the conditional independence assumption in BNs as well as the assumptions that the samples are independent, we can get the equation as follows:

$$\begin{aligned} L_D(\theta) &= \log\left\{\prod_{m=1}^{N} p(x_1[m],\ldots,x_n[m]:\theta)\right\} \\ &= \log\left\{\prod_{i=1}^{n}\prod_{m=1}^{N} p(x_i[m]|pa_i[x_i(m)]:\theta)\right\} \end{aligned} \tag{2}$$

where $pa_i[x_i(m)]$ indicates the ith parent of $x_i(m)$. With the MLE (Maximum Likelihood Estimation) method, we can get the parameter θ^* as follows:

$$\theta^*=\arg\max_{\theta} L_D(\theta) \tag{3}$$

However, when the data D is incomplete, Eq. (2) cannot be directly applied anymore. A common method is the EM algorithm [6]. Let $Y=\{Y_1,Y_2,\ldots,Y_N\}$, which is observed data; $Z=\{Z_1,Z_2,\ldots,Z_N\}$, which is missing data; and $D_l=Y_l\cup Z_l$. The EM algorithm starts with some initial guess at the maximum likelihood parameter, $\theta^{(0)}$, and then proceeds to iteratively generate successive estimates, $\theta^{(1)}$, $\theta^{(2)}$,..., by repeatedly applying the Expectation-step and Maximization-step, for $t=1,2,\ldots$

E-step: computes the conditional expectation of the log-likelihood function given the observed data Y and the current parameter $\theta^{(t)}$

$$Q(\theta|\theta^{(t)})=E_{\theta^{(t)}}[\log p(D|\theta)|\theta^{(t)},Y] \tag{4}$$

M-step: finds a new parameter $\theta^{(t)}$ which maximizes the expected log-likelihood under the assumption that the distribution found in the E-step is correct

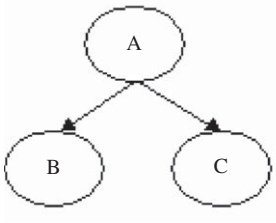
$$\theta^{(t+1)}=\arg\max_{\theta} Q(\theta|\theta^{(t)}) \tag{5}$$

Each iteration is guaranteed to increase the likelihood, and finally the algorithm converges to a local maximum of the likelihood function. However, when there are multiple hidden nodes or when a large amount of data are missing, EM method easily gets stuck in a local maximum. Next, we show how to incorporate domain knowledge into the learning process in order to reduce search space and to avoid local maxima.

3.2. Qualitative constraints with confidence

In many real-world applications, domain experts usually have valuable information about model parameters. We consider two types of constraints: type-I is about the range of a parameter; and type-II is about the relative relationships ($>,<,=$) between different parameters. One of our goals is to make the constraints as simple as possible, so that experts can easily formalize their knowledge into these constraints.

The range of a parameter allows domain experts to specify an upper bound and a lower bound for the parameter, instead of defining specific values. Fig. 1 shows a very simple BN and we assume all the nodes have binary states. The table in the right is the conditional probability table of the node B, indicating $P(B|A)$, which can be described by four parameters θ_{B00}, θ_{B01}, θ_{B10}, θ_{B11}, where $\theta_{B00}+\theta_{B01}=1$ and $\theta_{B10}+\theta_{B11}=1$. The domain experts may not know the specific



	$p(B\|A)$	
A	B	
	0	1
0	θ_{B00}	θ_{B01}
1	θ_{B10}	θ_{B11}

Fig. 1. A simple Bayesian network example and its conditional probability table.

values of θ_{B00} and θ_{B10}, but they can set the ranges for θ_{B00} and θ_{B10} as $0.3 < \theta_{B00} < 0.5$, $0.6 < \theta_{B10} < 0.9$, respectively.

In addition to assessing the ranges of parameters, the domain experts may also know the relative relationships between some parameters. For each type-II constraint, if the two associated parameters are in the same CPT of a node, we call it an inner-relationship constraint; if the two parameters come from the CPTs of different nodes, we call it an outer-relationship constraint. For example, if the observation of $A=0$ increases the posterior of $B=0$ the most, we could say that $p(B=0|A=0) > p(B=0|A=1)$, in other words, $\theta_{B00} > \theta_{B10}$. Obviously, the constraint of $\theta_{B00} > \theta_{B10}$ defines an inner-relationship. On the other hand, if the observation of $A=0$ increases the posterior of $B=0$ more than the observation of $A=0$ increasing the posterior of $C=0$, we can get the constraint of $\theta_{B00} > \theta_{C00}$, which defines an outer-relationship since B and C are from two different CPTs.

In real-world applications, there are always such constraints that can be found by domain experts. For example, assume we use a BN to model human state and its symptoms such as blinking rate, head tilt rate, eye gaze distribution, etc. Some symptom may be a stronger indicator of a particular human state than another symptom. This kind of relationship can be captured by the type-II constraints in the BN. They can often be identified either through subjective experience or through a sensitivity analysis. These constraints look simple, but are very important for the hidden nodes, where no data are available. By adding these constraints into learning, the domain knowledge can be well utilized to obtain parameters that meet the requirements of real-world applications.

Now we formally define the two types of constraints. Let A be the set that includes the parameters whose ranges are known based on the domain knowledge. For each $\theta_{ijk} \in A$, we define the range as $l_{ijk} \leqslant \theta_{ijk} \leqslant u_{ijk}$. Obviously, $l_{ijk} \geqslant 0$, and $u_{ijk} \leqslant 1$. Let B be the set that includes the parameters whose relative relationships are known based on the domain knowledge. For each $\theta_{ijk}, \theta_{i'j'k'} \in B$, we have $\theta_{ijk} > \theta_{i'j'k'}$, and/or, $\theta_{ijk} = \theta_{i'j'k'}$, where $i \neq i'$, or $j \neq j'$, or $k \neq k'$.

However, the domain knowledge may not be reliable all the time. To account for it, we associate confidence levels λ_{ijk}, $\lambda_{ijk}^{i'j'k'}$ to each constraint in the sets A and B, respectively. The value of each λ is between 0 and 1. If a domain expert is very confident with a constraint, the corresponding value of λ is 1; otherwise, λ is less than 1 but larger than 0.

3.3. Parameter learning with uncertain qualitative constraints

Now our goal is to find the optimal parameter $\hat{\theta}$ that maximizes the log-likelihood $L_D(\theta)$ given the three constraints as below:

$$\begin{aligned}
&\text{Maximize} \quad L_D(\theta)\\
&\text{Subject to} \quad \sum_k \theta_{ijk} = 1\\
&\qquad 1 \leqslant i \leqslant n, \quad 1 \leqslant j \leqslant q_i, \quad 1 \leqslant k \leqslant q_i\\
&\qquad l_{ijk} \leqslant \theta_{ijk} \leqslant u_{ijk}, \quad \theta_{ijk} \in A\\
&\qquad \theta_{ijk} \geqslant \theta_{i'j'k'}, \theta_{ijk}, \theta_{i'j'k'} \in B
\end{aligned} \tag{6}$$

The above equation shows a constrained optimization problem. For each inequality constraint, we define the following penalty functions:

$$g'(\theta_{ijk}) = [\theta_{ijk} - l_{ijk}]^-, \quad \forall \theta_{ijk} \in A \tag{7}$$

$$g''(\theta_{ijk}) = [u_{ijk} - \theta_{ijk}]^-, \quad \forall \theta_{ijk} \in A \tag{8}$$

$$g'''(\theta_{ijk}, \theta_{i'j'k'}) = [\theta_{ijk} - \theta_{i'j'k'}]^-, \quad \forall \theta_{ijk}, \theta_{i'j'k'} \in B \tag{9}$$

where $[x]^- = \max(0, -x)$.

Therefore, we can rephrase Eq. (6) as follows:

$$\begin{aligned}
\text{Maximize} \quad & J(\theta) = L_D(\theta) - \frac{w_1}{2} \sum_{\theta_{ijk} \in A} \lambda_{ijk} [(g'(\theta_{ijk}))^2 + (g''(\theta_{ijk}))^2]\\
& - \frac{w_2}{2} \sum_{\theta_{ijk}, \theta_{i'j'k'} \in B} \lambda_{ijk}^{i'j'k'} (g'''(\theta_{ijk}, \theta_{i'j'k'}))^2\\
\text{Subject to} \quad & \sum_k \theta_{ijk} = 1
\end{aligned} \tag{10}$$

where w_i is the penalty weight, which is decided empirically. Obviously, the penalty varies with the confidence level for each constraint.

In order to solve the problem, first, we eliminate the constraint $\sum_k \theta_{ijk} = 1$ by reparameterizing θ_{ijk}, so that the new parameters automatically respect the constraint on θ_{ijk} no matter what their values are. We define a new parameter β_{ijk} so that

$$\theta_{ijk} \equiv \frac{\exp(\beta_{ijk})}{\sum_{k'=1}^{r_i} \exp(\beta_{ijk'})} \tag{11}$$

In this way, a local maximum w.r.t. to β_{ijk} is also a local maximum w.r.t. θ_{ijk}, and vice versa. Most importantly, the constraint is automatically satisfied.

In the next step, we need to compute the derivative of $J(\theta)$ w.r.t. β. Based on [24], $\nabla_{\theta_{ijk}} L_D(\theta)$ can be expressed as follows:

$$\begin{aligned}
\nabla_{\theta_{ijk}} L_D(\theta) &= \frac{\partial \ln \prod_{l=1}^N p(D_l|\theta)}{\partial \theta_{ijk}}\\
&= \sum_{l=1}^N \frac{\partial \ln p(D_l|\theta)}{\partial \theta_{ijk}}\\
&= \sum_{l=1}^N \frac{\partial \ln p(D_l|\theta)/\partial \theta_{ijk}}{p(D_l|\theta)}
\end{aligned} \tag{12}$$

where

$$\begin{aligned}
\frac{\partial \ln p(D_l|\theta)/\partial \theta_{ijk}}{p(D_l|\theta)} &= \frac{\frac{\partial}{\partial \theta_{ijk}} \sum_{j',k'} p(D_l|x_i^{k'}, pa_i^{j'}, \theta) p(x_i^{k'}, pa_i^{j'}|\theta)}{p(D_l|\theta)}\\
&= \frac{\frac{\partial}{\partial \theta_{ijk}} \sum_{j',k'} p(D_l|x_i^{k'}, pa_i^{j'}, \theta) p(x_i^{k'}|pa_i^{j'}, \theta) p(pa_i^{j'}|\theta)}{p(D_l|\theta)}\\
&= \frac{p(D_l|x_i^k, pa_i^j, \theta) p(pa_i^j|\theta)}{p(D_l|\theta)}\\
&= \frac{p(x_i^k, pa_i^j|D_l, \theta) p(D_l|\theta) P(pa_i^j|\theta)}{p(x_i^k, pa_i^j|\theta) p(D_l|\theta)}\\
&= \frac{p(x_i^k, pa_i^j|D_l, \theta)}{p(x_i^k, pa_i^j|pa_i^j, \theta)}\\
&= \frac{p(x_i^k, pa_i^j|D_l, \theta)}{\theta_{ijk}}
\end{aligned} \tag{13}$$

By combining Eqs. (12) and (13), we obtain the equation as below:

$$\nabla_{\theta_{ijk}} L_D(\theta) = \frac{\sum_{l=1}^{N} p(x_i^k, pa_i^j | D_l, \theta)}{\theta_{ijk}} \tag{14}$$

Therefore, based on the chain rule,

$$\begin{aligned}\nabla_{\beta_{ijk}} L_D(\theta) &= \frac{\partial L_D(\theta)}{\partial \theta_{ijk}} \frac{\partial \theta_{ijk}}{\partial \beta_{ijk}} \\ &= \nabla_{\theta_{ijk}} L_D(\theta)(\theta_{ijk} - \theta_{ijk}^2) \\ &= \sum_{l=1}^{N} p(x_i^k, pa_i^j | D_l, \theta)(1 - \theta_{ijk})\end{aligned} \tag{15}$$

Similarly, for $g'(\theta_{ijk})$, $g''(\theta_{ijk})$, and $g'''(\theta_{ijk})$, the derivatives are as follows:

$$\nabla_{\beta_{ijk}} g'(\theta_{ijk}) = \begin{cases} \theta_{ijk}^2 - \theta_{ijk} & \text{if } \theta_{ijk} \leqslant l_{ijk} \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

$$\nabla_{\beta_{ijk}} g''(\theta_{ijk}) = \begin{cases} \theta_{ijk} - \theta_{ijk}^2 & \text{if } \theta_{ijk} \geqslant u_{ijk} \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

$$\nabla_{\beta_{ijk}} g'''(\theta_{ijk}, \theta_{i'j'k'}) = \begin{cases} \theta_{ijk}^2 - \theta_{ijk} & \text{if } \theta_{ijk} \leqslant \theta_{i'j'k'},\ i \neq i' \text{ or } j \neq j' \\ \theta_{ijk}^2 - \theta_{ijk} - \theta_{ijk}\theta_{i'j'k'} & \text{if } \theta_{ijk} \leqslant \theta_{i'j'k'},\ i = i',\ j = j' \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

Table 1
A constrained EM (CEM) learning algorithm.

Repeat until it converges
  Step 1: E-step to compute the conditional expectation of the log-likelihood function based on Eq. (4);
  Step 2: M-step to find the parameter θ' that maximizes the expected log-likelihood based on Eq. (5);
  Step 3: Perform the following optimization procedure based on the gradient-descent method:
    $\theta^t = \theta'$; map θ^t to β^t based on Eq. (11)
    Repeat until $\Delta\beta^t \simeq 0$
      $\Delta\theta^t = 0$
      for each variable i, parent configuration j, value k
        for each $D_l \in D$
          $\Delta\theta_{ijk}^{t+1} = \Delta\theta_{ijk}^t + p(x_i^k, pa_i^j | D_l, \theta_t)$
      $\Delta\beta_{ijk}^{t+1} = \Delta\theta_{ijk}^{t+1}(1 - \theta_{ijk}) + K$
      (K represents the last three terms in Eq. (19))
    $\beta^{t+1} = \beta^t + \Delta\beta^{t+1}$
    map β^{t+1} to θ^{t+1} based on Eq. (11)
    $\theta^t = \theta^{t+1}$
  Go to Step 1
Return θ^t

Therefore, the derivative of $J(\theta_{ijk})$ w.r.t. β_{ijk} is as we can see here:

$$\begin{aligned}\nabla_{\beta_{ijk}} J(\theta_{ijk}) = &\sum_{l=1}^{N} p(x_i^k, pa_i^j | D_l, \theta)(1 - \theta_{ijk}) \\ &- w_1 \lambda_{ijk}[g'(\theta_{ijk}) \nabla_{\beta_{ijk}} g'(\theta_{ijk}) + g''(\theta_{ijk}) \nabla_{\beta_{ijk}} g''(\theta_{ijk})] \\ &- w_2 \sum_{B^+(\theta_{ijk})} [\lambda_{ijk}^{i'j'k'} g'''(\theta_{ijk}, \theta_{i'j'k'}) \nabla_{\beta_{ijk}} g'''(\theta_{ijk}, \theta_{i'j'k'})] \\ &+ w_2 \sum_{B^-(\theta_{ijk})} [\lambda_{ijk}^{i'j'k'} g'''(\theta_{i'j'k'}, \theta_{ijk}) \nabla_{\beta_{ijk}} g'''(\theta_{i'j'k'}, \theta_{ijk})]\end{aligned} \tag{19}$$

where $B^+(\theta_{ijk})$ is the set of the constraints whose first term is θ_{ijk}, while $B^-(\theta_{ijk})$ is the set of the constraints whose second term is θ_{ijk}. Both $B^+(\theta_{ijk})$ and $B^-(\theta_{ijk})$ belong to the set B.

Now, we are ready to present the constrained EM (CEM) learning algorithm as summarized in Table 1. The algorithm consists of three steps. The first two steps are the same as the E-step and M-step in the EM algorithm. In the third step, a gradient-based update is used to force the solutions to move towards the direction of reducing constraint violations.

4. Experiments

In this section, we compare our algorithm to the EM algorithm with synthetic data. We first describe how the testing data is generated, and then demonstrate the results in three scenarios: (1) varying the type-I constraints; (2) varying the type-II constraints; and (3) varying the number of training samples. Furthermore, in the next section, we will apply our algorithm to a real-world application.

4.1. Experiment design

In order to compare the CEM algorithm to the EM algorithm, we design the experiments as follows.

Generation of original BNs. Two BNs as shown in Fig. 2 are created in the experiments. Then 22 instances are created with the same structures as BN1 and BN2, respectively (11 for each), but with different parameters. Two BNs (called original BNs) are then used as the ground-truth for BN1 and BN2, respectively, and the others are to be learned.

Generation of constraints. For each BN, based on the true CPTs, type-I (the range of a parameter) and type-II (the relationship between different parameters) constraints are generated for the node sets A and B, where A and B vary in the experiments. Specifically, to generate type-I constraints for the nodes in A, for each true parameter θ_{ijk}, the lower bound is set as $(1 - r)\theta_{ijk}$, and the upper bound is set as $min(1, (1 + r)\theta_{ijk})$, where r is a ratio ($0 < r < 1$) and varies in the experiments. Type-II constraints can be divided into

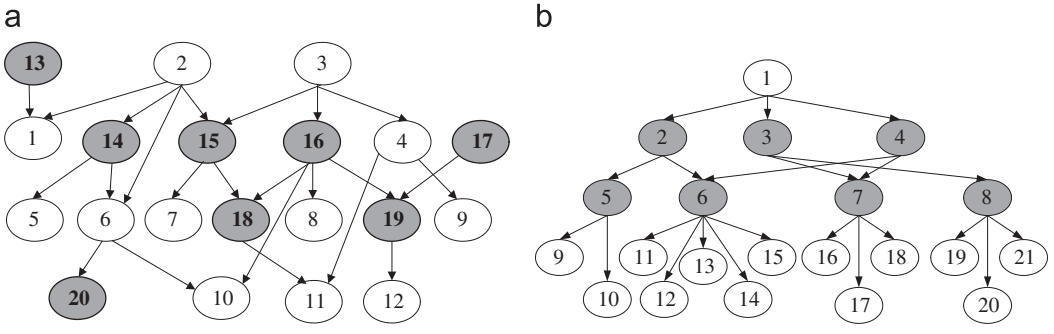


Fig. 2. Two BN examples: (a) BN1; (b) BN2. The shaded nodes are hidden nodes, the others are observable nodes.

two kinds: inner-relationship, which compares two parameters within the same CPT; outer-relationship, which compares two parameters in different CPTs. For example, in BN1, two parameters in the CPT of the node 18 can be associated with an inner-relationship constraint; while a parameter in the CPT of the node 18 and a parameter in the CPT of the node 19 can be associated with an outer-relationship constraint. Table 2 shows some examples of type-II constraints for BN1. For each parameter θ_{abcd} in the table, the first two numbers (ab) of the subscript represent the index of the node, and the third number (c) represents the index of the parent configurations, and the last number (d) represents the state index of the node.

Table 2
Some examples of type-II constraints for BN1.

Inner-relationship constraints	Outer-relationship constraints
Node 14: $\theta_{1411} > \theta_{1421}$	Node 13 and Node 17: $\theta_{1311} > \theta_{1711}$
Node 15: $\theta_{1511} > \theta_{1522}$	Node 14 and Node 15: $\theta_{1411} > \theta_{1531}$
Node 20: $\theta_{2011} > \theta_{2021}$	Node 18 and Node 19: $\theta_{1821} > \theta_{1941}$

Generation of training data. For each BN, 500 samples are generated based on the true parameters. The values of the hidden nodes are then removed from the generated samples. With the remaining samples, we then learn the 20 BNs with randomly assigned initial parameters, which are required to be different from the true parameters.

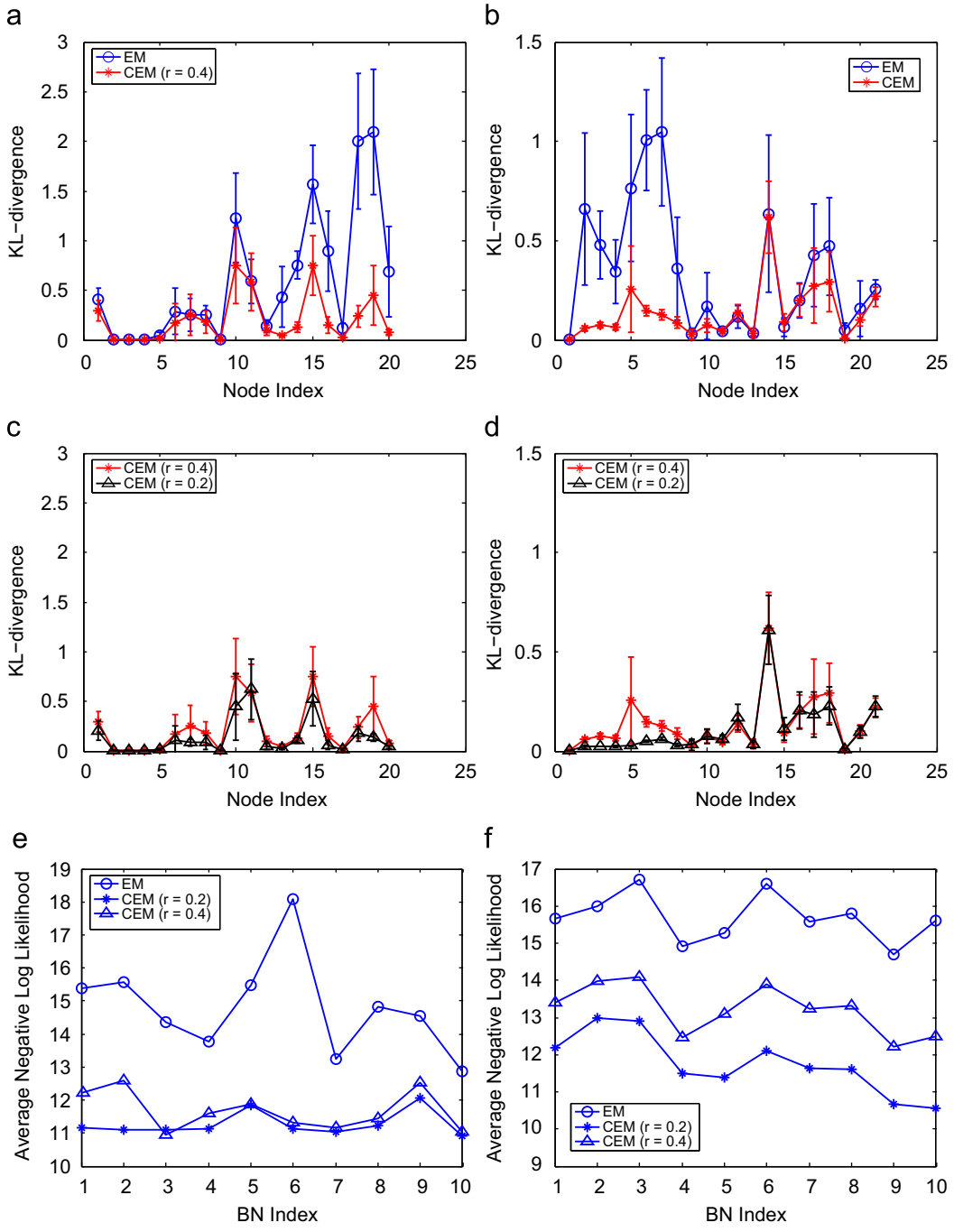


Fig. 3. Learning results vs. type-I constraints. The charts in the left are the results for BN1; and the charts in the right are the results for BN2. (a), (b) EM vs. CEM when $r = 0.4$; (c), (d) CEM when $r = 0.4$ and 0.2; and (e), (f) negative log-likelihood for different BNs.

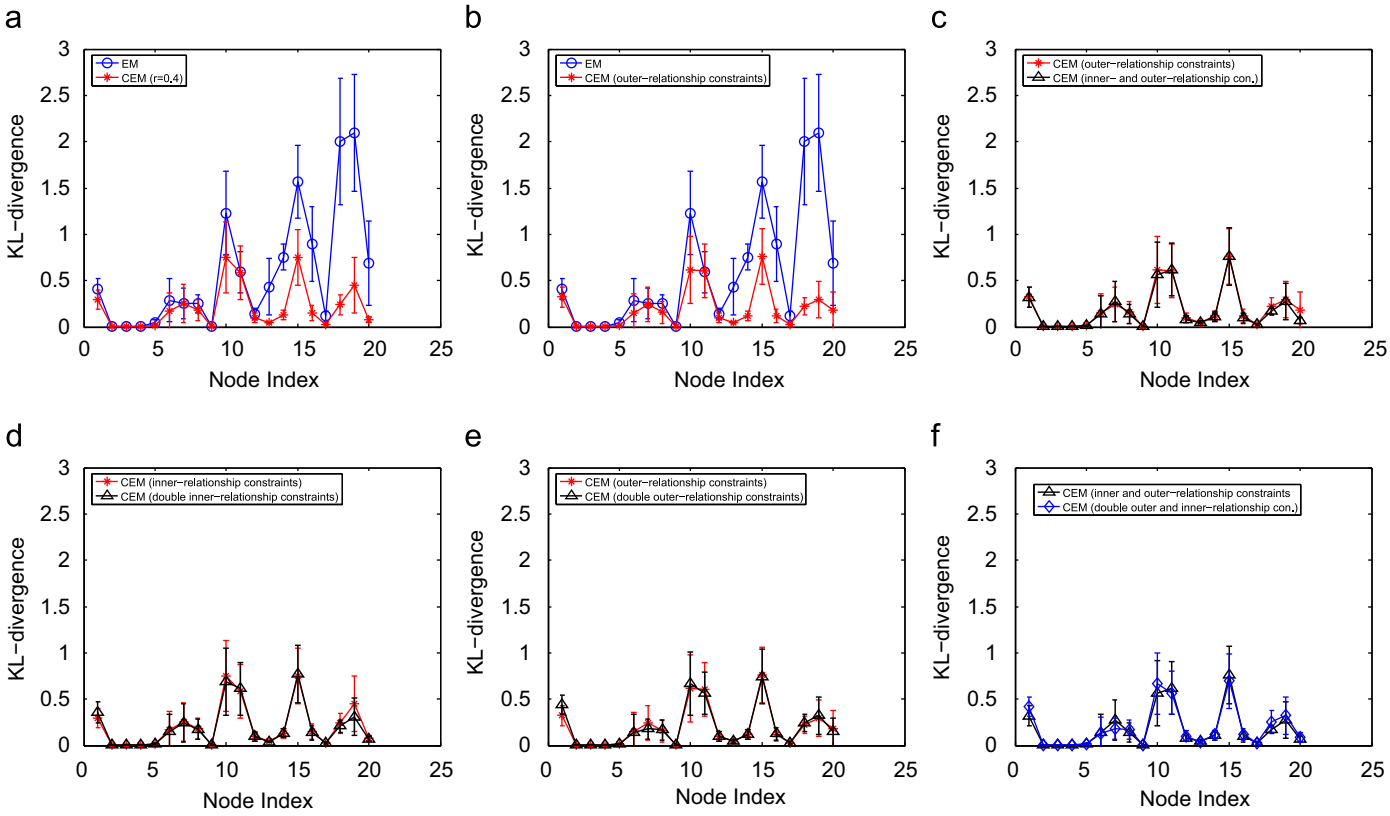


Fig. 4. Learning results vs. type-II constraints for BN1: (a) EM vs. CEM (eight inner-relationship constraints for the hidden nodes are used); (b) EM vs. CEM (eight outer-relationship constraints for the hidden nodes are used); (c) CEM (eight outer-relationship constraints are used) vs. CEM (eight outer-relationship and eight inner-relationship constraints are used); (d) eight and 16 inner-relationship constraints are used, respectively; (e) eight and 16 outer-relationships are used, respectively; and (f) CEM (eight outer-relationship and eight inner-relationship constraints are used) vs. CEM (16 inner-relationship and 16 outer-relationship constraints are used). BN2 has the similar results.

Evaluation of performance. Two criteria are used to evaluate the performance of the learning algorithms. The first one is to compare the estimated CPT of each node to the ground-truth CPT using the Kullback–Leibler (KL) divergence. The smaller the KL-divergence is, the closer the estimated CPT to the true CPT. The second criterion is the negative log-likelihood per sample. It evaluates how well a learned BN fits the testing data as a whole. To compute that, we first generate 500 testing samples from each original BN. Each of the learned BNs is then evaluated on these samples to get the average negative log-likelihood. Since it is negative log-likelihood, the smaller the value is, the better the learned BN fits the data.

4.2. Learning results vs. type-I constraints

In this section, we compare learning performance when type-I constraints vary while type-II constraints are fixed. Specifically, both the set A and B include only hidden nodes. For type-I constraints, r varies from 0.2 to 0.4. For type-II constraints, only the inner-relationships for two parameters in the CPT of each hidden node are used as constraints.

Fig. 3 illustrates the results. In Charts (a) through (d), the x-coordinate denotes the node index, and the y-coordinate denotes the KL-divergence. The median of each bar is the mean, and the height of the bar is the standard deviation, which are obtained from 10 BN instances. Charts (a) and (b) compare EM to CEM when $r = 0.4$. We can see that CEM achieves better results in both mean and standard deviation of KL-divergence than EM for both BNs. In BN1, for the hidden nodes, the average mean decreases from 1.0687 (EM) to 0.2337 (CEM, $r = 0.4$); the average standard deviation decreases from 0.7659 (EM) to 0.2219 (CEM, $r = 0.4$). In BN2, for the hidden nodes, the average mean decreases from 0.6657 (EM) to 0.1163 (CEM, $r = 0.4$); the average standard deviation decreases from 0.5614 (EM) to 0.0969 (CEM, $r = 0.4$). Specifically, as shown in Charts (a) and (c), for the hidden node 18 in BN1, the KL-divergence decreases from around 2.1 (EM) to 0.2 (CEM, $r = 0.2$); for the hidden node 19 in BN1, the KL-divergence decreases from around 2.2 (EM) to 0.2 (CEM, $r = 0.2$).

Charts (c) and (d) compare CEM when r varies. As r decreases from 0.4 to 0.2, the performance of CEM is further improved, especially for the hidden nodes. The negative log-likelihood further confirms that CEM performs better than EM, as shown in Charts (e) and (f), where the x-coordinate indicates BN index and the y-coordinate indicates the negative log-likelihood. The negative log-likelihood from CEM is smaller than that from EM.

4.3. Learning results vs. type-II constraints

In the second scenario, we change type-II constraints while fix the type-I constraints (r is set as 0.4). We observe the learning results in the following cases: (1) varying the number of inner-relationship constraints; (2) varying the number of outer-relationship constraints; and (3) combining inner-relationship constraints and outer-relationship constraints.

Fig. 4 illustrates the results in the three cases. Charts (a) and (b) compare EM to CEM when eight inner-relationship constraints and eight outer-relationship constraints are used, respectively. Obviously, CEM always performs better than EM. For example, in Chart (a), the average mean for the hidden nodes decreases from 1.0687 (EM) to 0.2337 (CEM), the average standard deviation decreases

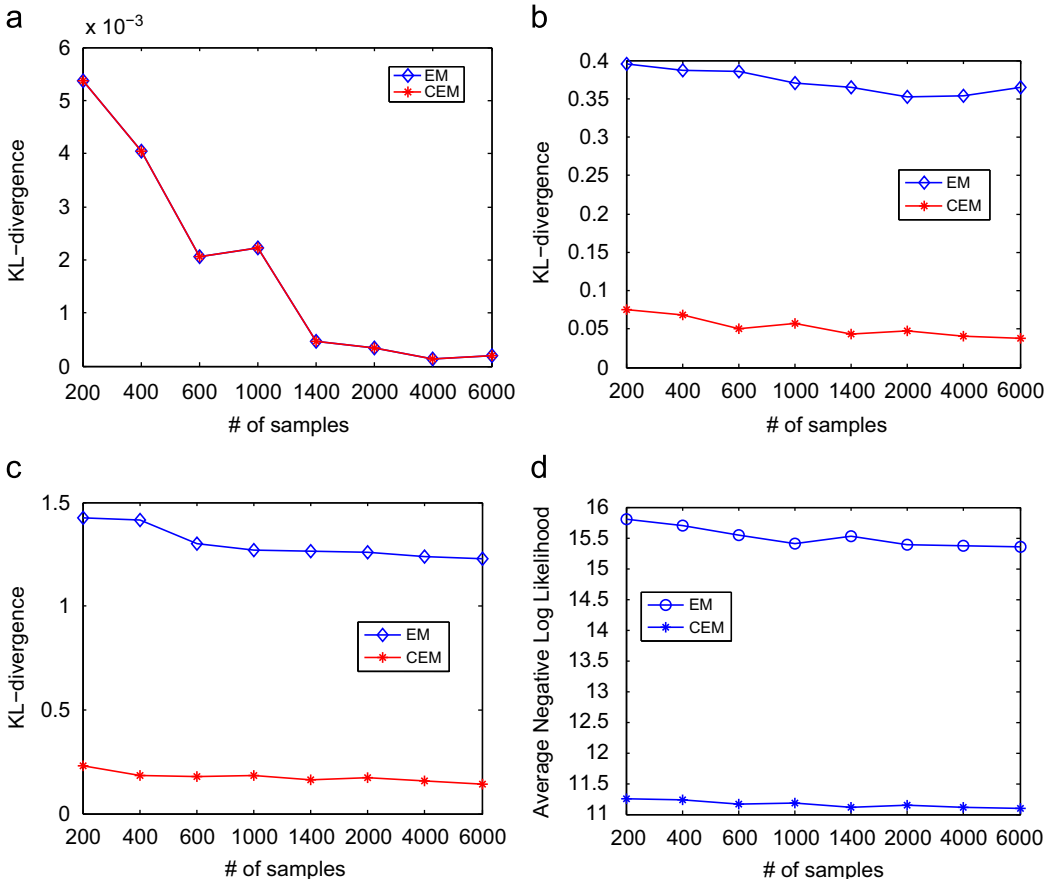


Fig. 5. Learning results vs. the number of training samples for BN1: (a) Average KL-divergence for the observable nodes whose parameters are independent from the hidden nodes (i.e., nodes 2, 3, 4, 9); (b) average KL-divergence for the other observable nodes; (c) average KL-divergence for the hidden nodes; and (d) negative log-likelihood. BN2 has the similar results.

from 0.7659 (EM) to 0.2219 (CEM); and in Chart (b), the average mean for the hidden nodes decreases from 1.0687 (EM) to 0.2214 (CEM), the average standard deviation decreases from 0.7659 (EM) to 0.2377 (CEM). Charts (c) through (f) show how the performance of CEM varies as different numbers of constraints are used. Chart (c) demonstrates that when both inner-relationship constraints and outer-relationship constraints are used, the performance is better than single-type constraints are used. The average mean for the hidden nodes decreases from 0.2214 to 0.1929, and the average standard deviation decreases from 0.2377 to 0.1763. Charts (d)–(f) show that the performance of CEM is improved slightly when we double the same types of constraints.

4.4. Learning results vs. training samples

We now demonstrate how the learning results vary with the number of training samples. In the experiments, we fix the constraints (r=0.2), and only eight inner-relationship constraints for the hidden nodes are used, but vary the number of training samples during learning. Fig. 5 demonstrates the results for BN1 only, since BN2 has the similar results. In order to observe how the training samples affect different types of nodes in BN1, we divide the nodes into three groups: group 1 includes the nodes (2, 3, 4, and 9) whose parameters are independent from the hidden nodes; group 2 includes the other observable nodes; and group 3 includes all the hidden nodes.

As shown in Chart (a) of Fig. 5, for the nodes in group 1, the KL-divergence decreases when the number of samples increases. And the KL-divergence is very small in all the cases even when there are only 200 training samples. Both EM and CEM return the same results since the data are complete for all those nodes. In both Charts (b) and (c), the KL-divergence decreases slightly when the number of training samples increases, while the KL-divergence of CEM is much smaller than that of EM. Especially for the hidden nodes, even when the number of the training sample is 6000, the KL-divergence for the hidden nodes is only slightly smaller than the KL-divergence when the number of training sample is 200. This is because the information about the hidden nodes rarely varies as the total number of training samples increases. Therefore, in order to improve the learning results for the hidden nodes, domain knowledge is more important than training samples.

5. A case study

In this section, we apply our algorithm to a real-world application in computer vision: facial action unit (AU) recognition.

5.1. A Bayesian network for facial action unit modeling and recognition

In recent years, a variety of approaches are proposed to recognize facial expressions. Besides recognizing six basic facial expressions directly, techniques have also been developed to automatically recognize facial action units. According to the Facial Action Unit System (FACS) by Ekman [26], each AU is related to the contraction of a specific set of facial muscles. FACS has been demonstrated to be a powerful means for representing and characterizing a large number

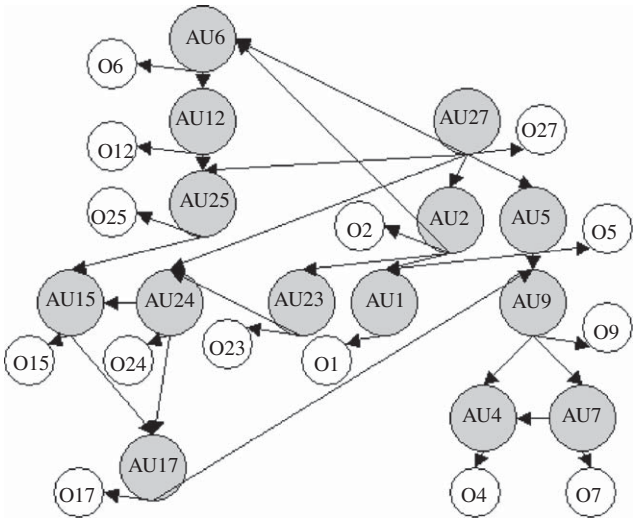


Fig. 6. A Bayesian network for AU modeling. We adapted it from [27]. The unshaded nodes are measurement nodes.

Table 3
A list of action units.

AU1  Inner brow raiser	AU2  Outer brow raiser	AU4  Brow Lowerer	AU5  Upper lid raiser	AU6  Cheek raiser
AU7  Lid tighten	AU9  Nose wrinkle	AU12  Lip corner puller	AU15  Lip corner depressor	AU17  Chin raiser
AU23  Lip tighten	AU24  Lip presser	AU25  Lips part	AU27  Mouth stretch	

of facial expressions through the combination of only a small set of AUs.

Current techniques for recognizing AUs are mostly based on computer vision techniques. But due to the richness, ambiguity, and dynamic nature of facial actions, as well as the image uncertainty and individual difference, it is difficult to recognize each AU individually by using only computer vision techniques. Computer vision techniques should be combined with additional sematic information to achieve a more robust and consistent AU recognition. Fortunately, there are some inherent relationships among AUs as described in the FACS manual [26]. For example, the alternative rules provided in the FACS manual describe the mutual exclusive relationship among AUs. Furthermore, the FACS also includes the co-occurrence rules in their old version, which were "designed to make scoring more deterministic and conservative, and more reliable" [26].

Therefore, instead of recognizing each AU alone, a Bayesian network can be used to model the probabilistic relationships among different AUs, and given the BN model, AU recognition can be performed through a probabilistic inference [27]. Fig. 6 illustrates such a BN to model 14 AUs, and Table 3 summarizes a list of the 14 AUs and their meanings. They are adapted from [27]. Such a model is capable of representing the relationships among different AUs in a coherent and unified hierarchical structure, accounting for uncertainties in the AU recognition process with conditional dependence links, and providing principled inference and learning techniques to systematically combine the domain information and statistical information extracted from the data.

To incorporate the AU recognition results from a computer vision technique, in the BN model, each AU is connected with a measurement node (unshaded node), which encodes the measurement obtained from computer vision techniques. For AU measurement, we employ a technique similar to the one described in [28]. The output of the technique is a score for each AU, which is subsequently discredited to produce a value for a corresponding AU measurement node.

5.2. AU model parameter learning

Given the BN structure in Fig. 6 and the AU measurements, we then need parameterize the BN model before AU inference can commence. For this, we need training data. As defined before, a complete training sample requires the true AU label for each AU node and the measurement for each meresman node. However, manually labeling AUs is usually time consuming and difficult. Moreover, the labeling process is highly subjective, therefore prone to human errors. In addition, some AU events rarely happen in the collected data. Therefore, the training data could be incomplete, biased, or spare for certain AUs. We thus apply our algorithm to learn the BN parameters using only the AU measurements and some domain knowledge, without any AU labeling.

We first generate constraints. For type I constraints, domain experts are consulated to specify the approximate ranges for most of the parameters. Type II constraints are also constructed from domain specific knowledge. Specifically, for each measurement node, since the measurement accuracy for each AU varies, depending the computer vision technique used as well as on the difficulty of the AU, we can rank the measurements by their accuracy and then translate such a ranking into the outer-relationships between the corresponding measurement nodes. For example, the computer vision technique usually performs better in recognition of AU2 (outer brow raiser) than AU23 (lip tighten), hence we can get constraints like $p(O2=0|AU2=0) > p(O23=0|AU23=0)$, $p(O2=1|AU2=1) > p(O23=1|AU23=1)$, where 0 means an AU is absent, and 1 means an AU is present.

More type-II constraints can be obtained based on the properties of different AUs. For example, for AU6 (cheek raiser) and AU12 (lip corner puller), the probability of AU12 being absent if AU6 is absent, is smaller than the probability of AU6 being present if AU12 is present, i.e., $p(AU12=0|AU6=0) < p(AU12=1|AU6=1)$. For AU1 (inner brow raiser), the influence of AU2 (outer brow raiser) on AU1 is larger than the influence of AU5 (upper lid raiser) on AU1, i.e. $p(AU1=1|AU2=1, AU5=0) > p(AU1=1|AU2=0, AU5=1)$. Overall, we generate one type-I constraint for each parameter, and about 28 type-II constraints for all the parameters. Of course, the number of constraints depends on the application and the domain knowledge available for the application.

5.3. AU recognition results

We use 8000 images collected from Cohan and Kanade's DFAT-504 database [29], where 80% are used for training and 20% data are used for testing. We first use MLE to learn parameters from the complete data, which consists both the true AU labels and the AU measurements. Then, we use the EM and CEM algorithms to learn parameters from the incomplete data, which only includes the AU measurements.

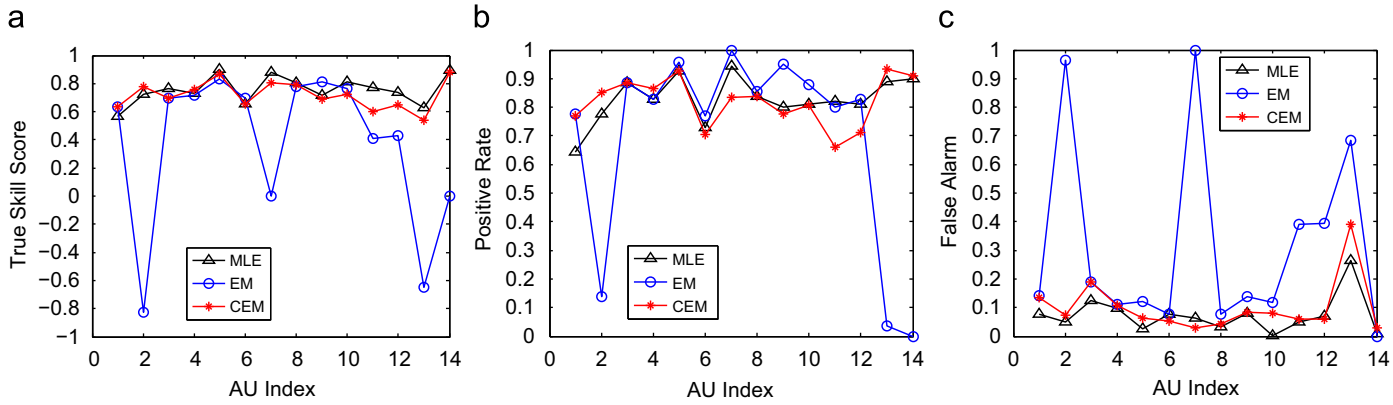


Fig. 7. Comparison of average AU recognition results using the BNs learned from MLE, EM, and CEM, respectively: (a) true skill score; (b) positive rate; and (c) false alarm. MLE uses complete training data, while EM and CEM use incomplete data that only include AU measurements.

Fig. 7 compares the AU recognition results with BNs learned from MLE, EM, and CEM in terms of true skill score (the difference between the positive rate and the false alarm), positive rate, and false alarm. CEM performs similarly to MLE (based on complete data), but much better than EM. The average true skill score is only 0.37 for EM, 0.72 for CEM, and 0.76 for MLE. The positive rate increases from 0.69 (EM) to 0.82 (CEM), and the false alarm decreases from 0.32 (EM) to 0.1 (CEM). For EM, some AUs totally fail, such as AU2, AU7, and AU23. But CEM has a fair performance for all the AUs even it is learned from the unlabeled data. This again shows the importance of domain knowledge. CEM is able to fully utilize the domain knowledge for automatic parameter learning. Compared to MLE that is based on the labeled data, the CEM has comparable performance but without using any labeled AUs. This is indeed very encouraging. This may represent a significant step forward in machine learning in general and BN learning in particular.

6. Conclusion and future work

When a large amount of data are missing, or when multiple hidden nodes exist, learning parameters in Bayesian networks becomes extremely difficult. The learning algorithms are required to operate in a high-dimensional search space and could easily get trapped among copious local maxima. We thus present a constrained EM algorithm to learn Bayesian network parameters when a large amount of data are missing in the training data. The algorithm fully utilizes certain qualitative domain knowledge to regularize the otherwise ill-posed problem, limit the search space, and avoid local maxima. Compared with the quantitative domain knowledge such as prior probability distribution typically used by the existing methods, the qualitative domain knowledge is local (only concerned with some parameters), easy to specify, and does not need strong assumption.

For many computer vision and pattern recognition problem, data is often hard to acquire and the model becomes increasingly complex. It, therefore, becomes increasingly important to incorporate human knowledge into the otherwise ill-posed learning process. Our method can solicit simple yet effective qualitative constraints from human experts, and then systematically incorporate them into the learning process. The improvement in learning performance is significant. Both the experiments from the synthetic data and real data for facial action recognition demonstrate that our algorithm improves the accuracy of the learned parameters significantly over the traditional EM algorithm.

The domain knowledge in the current learning algorithm was formalized by two simple constraints: a range of a parameter, and relative relationships between different parameters. Although they are very useful, it is possible to introduce more types of constraints into learning, such as the relationships between the sum of several parameters, parameter sharing, etc. More constraints will help further reduce the search space, although they may not be easy for domain experts to specify.

Furthermore, we assumed model structures are known and thus only focused on learning model parameters. However, in many applications, the structure could be unknown, or it is too difficult for domain experts to manually construct a complete structure. Learning the BN structure is therefore also necessary. Most current approaches to BN structure learning assume generic prior probabilities on graph structures, typically encoding a sparseness bias but otherwise expecting no regularities in the learned structures. We believe that the domain knowledge about model parameters can also help in learning the model structure.

References

[1] E. Delage, H. Lee, A. Ng, A dynamic Bayesian network model for autonomous 3d reconstruction from a single indoor image, in: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, 2006.
[2] J.M. Pena, J. Bjorkegren, J. Tegner, Growing Bayesian network models of gene networks from seed genes, Bioinformatics (2005) 224–229.
[3] L.M. de Campos, J.M. Fernández-Luna, J.F. Huete, Bayesian networks and information retrieval: an introduction to the special issue, Information Processing and Management (2004) 727–733.
[4] Y. Zhang, Q. Ji, Active and dynamic information fusion for facial expression understanding from image sequence, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (5) (2005) 699–714.
[5] N. Friedman, M. Goldszmidt, D. Heckerman, S. Russell, Where is the impact of Bayesian networks in learning? in: International Joint Conference on Artificial Intelligence, 1997.
[6] A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, The Royal Statistical Society Series B 39 (1977) 1–38.
[7] S. Geman, D. Geman, Stochastic relaxation, Gibbs distribution and the Bayesian restoration of images, IEEE Transactions on Pattern Analysis and Machine Intelligence 6 (1984) 721–741.
[8] M. Jaeger, The AI&M procedure for learning from incomplete data, in: Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence, 2006, pp. 225–232.
[9] M. Ramoni, P. Sebastiani, Robust learning with missing data, Machine Learning 45 (2) (2001) 147–170.
[10] R.G. Cowell, Parameter learning from incomplete data using maximum entropy I: principles, Statistical Research Report, vol. 21, 1999.
[11] R.G. Cowell, Parameter learning from incomplete data using maximum entropy II: application to Bayesian networks, Statistical Research Report, vol. 21, 1999.
[12] G. Elidan, N. Friedman, The information bottleneck EM algorithm, in: Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence, 2003, pp. 200–209.
[13] G. Elidan, M. Ninio, N. Friedman, D. Schuurmans, Data perturbation for escaping local maxima in learning, in: Proceedings of the 18th National Conference on Artificial Intelligence, 2002, pp. 132–139.

[14] B. Thiesson, Accelerated quantification of Bayesian networks with incomplete data, in: Proceedings of the First International Conference on Knowledge Discovery and Data Mining, 1995, pp. 306–311.
[15] E. Bauer, D. Koller, Y. Singer, Update rules for parameter estimation in Bayesian networks, in: Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence, 1997, pp. 3–13.
[16] W.L. Buntine, Operations for learning with graphical models, Artificial Intelligence Research 2 (1994) 159–225.
[17] S.T. Lauritzen, The EM algorithm for graphical association models with missing data, Computational Statistics and Data Analysis 19 (1995) 191–201.
[18] R.S. Niculescu, T.M. Mitchell, R.B. Rao, A theoretical framework for learning Bayesian networks with parameter inequality constraints, in: Proceedings of the 20th International Joint Conference on Artificial Intelligence, 2007.
[19] M.J. Druzdzel, L.C. van der Gaag, Elicitation of probabilities for belief networks: combining qualitative and quantitative information, in: Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence, 1995, pp. 141–148.
[20] F. Wittig, A. Jameson, Exploiting qualitative knowledge in the learning of conditional probabilities of Bayesian networks, in: Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence, 2000, pp. 644–652.
[21] E.E. Altendorf, A.C. Restificar, T.G. Dietterich, Learning from sparse data by exploiting monotonicity constraints, in: Proceedings of the 21th Conference on Uncertainty in Artificial Intelligence, 2005, pp. 18–26.
[22] A. Feelders, L. var der Gaag, Learning Bayesian network parameters with prior knowledge about context-specific qualitative influences, in: Proceedings of the 21th Conference on Uncertainty in Artificial Intelligence, 2005, pp. 193–20.
[23] S. Russell, J. Binder, D. Koller, K. Kanazawa, Local learning in probabilistic networks with hidden variables, in; Proceedings of the 14th International Joint Conference on Artificial Intelligence, 1995, pp. 1146–1152.
[24] J. Binder, D. Koller, S. Russell, K. Kanazawa, Adaptive probabilistic networks with hidden variables, Machine Learning (1997) 213–244.
[25] A. Feelders, L. var der Gaag, Learning Bayesian network parameters under order constraints, International Journal of Approximate Reasoning 42 (2006) 37–53.
[26] P. Ekman, W. Friesen, Facial Action Coding System: A Technique for the Measurement of Facial Movement, Consulting Psychologists Press, Palo Alto, CA, 1978.
[27] Y. Tong, W. Liao, Q. Ji, Inferring facial action units with causal relations, in: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, 2006.
[28] M.S. Bartlett, G. Littlewort, M. Frank, C. Lainscsek, I. Fasel, J. Movellan, Recognizing facial expression: machine learning and application to spontaneous behavior, Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition 2 (2005) 568–573.
[29] T. Kanade, J.F. Cohn, Y. Tian, Comprehensive database for facial expression analysis, in: Proceedings of FG00, 2000.

About the Author—WENHUI LIAO received the PhD degree from the Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, New York, in 2006. Her areas of research include probabilistic graphical models, information fusion, computer vision, and natural language processing. She is currently a research scientist at R&D of Thomson-Reuters Corporation.

About the Author—QIANG JI received the PhD degree in electrical engineering from the University of Washington in 1998. He is currently an associate professor in the Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute (RPI), Troy, New York. Prior to joining RPI in 2001, he was an assistant professor in the Department of Computer Science, University of Nevada, Reno. He also held research and visiting positions with Carnegie Mellon University, Western Research, and the US Air Force Research Laboratory. His research interests include computer vision, probabilistic reasoning with Bayesian networks for decision making and information fusion, human computer interaction, pattern recognition, and robotics. He has published more than 100 papers in peer-reviewed journals and conferences. His research has been funded by local and federal government agencies including the US National Science Foundation (NSF), the US National Institute of Health (NIH), the US Air Force Office of Scientific Research (AFOSR), the US Office of Naval Research (ONR), DARPA, and the US Army Research Office (ARO) and by private companies including Boeing and Honda. He is a senior member of the IEEE.