Irrelevance and Parameter Learning in Bayesian Networks

Nevin Lianwen Zhang
Department of Computer Science, University of Science & Technology, Hong Kong

Abstract

It is possible to learn the parameters of a given Bayesian network structure from data because those parameters influence the probability of observing the data. However, some of the parameters are irrelevant to the probability of observing a particular data case. This paper shows how such irrelevancies can be exploited to speedup various algorithms for parameter learning in Bayesian networks.

Keywords: Bayesian networks, parameter learning, irrelevance, efficiency.

1 Introduction

Of concern to this paper are problems of learning relationships among variables from data. Solution of such a problem is usually divided into two steps: (1) selecting a model for the relationships and (2) estimating the parameters in the model using data. Many models exist. There are statistical models such as regression and loglinear models; and there are AI models such as decision trees and artificial neural networks (ANN). Recently, there have been growing interests, from both statisticians and AI researchers, in learning with a model called Bayesian networks.

Bayesian networks (Pearl 1988, also called belief networks or probabilistic networks) are directed acyclic graphs, where each node represents a random variable and is attached with a conditional probability of the node given its parents. They have been widely accepted as a suitable knowledge representation framework for reasoning and decision under uncertainty.

In applications, Bayesian networks are usually constructed from experts' knowledge. Knowledge acquisition, especially assessment of numerical probabilities, is usually difficult and unreliable. There is a need to refine Bayesian networks using data (Spiegelhalter *et al* 1993), and even to automatically construct Bayesian networks from data (Heckerman 1995). This is one cause for the growing interests in learning Bayesian networks.

A second cause is the realization that Bayesian network learning can potentially rival (Russell *et al* 1995) or complement (Jordan 1995) neural network learning as a general learning paradigm. Since arcs in Bayesian networks can be interpreted as representing causal relationships, experts' qualitative knowledge, which is often readily available and reliable (e.g. it can be from a domain textbook), can be used to aid model selection and reduce the amount of training data. Also, parameters in a Bayesian network have clear semantics, while weights in ANNs do not in general.

There is one other advantage of Bayesian network learning that has received little attention up to now. Suppose a Bayesian network structure (model) has been selected and we want to estimate the parameters from data. This is possible because the parameters influence the probability of observing the data. For a particular data case, however, there is, due to the given network structure, a subset of parameter that are irrelevant to the probability of observing the data case. Consequently, the data case should not contribute anything to the estimation of those parameters. Such irrelevancies can be exploited to speedup learning.

Exploiting irrelevancies are especially important when there is a large number of variables in the model and there are multiple data sets, each concerning a different subset of variables. In such a case, the number of parameters that are relevant to a data case can be much smaller the total number of parameters.

This paper examines three types of algorithms for parameter learning in a given Bayesian network model, namely the EM algorithm (Lauritzen 1995), gradient descent (Russell *et al* 1995), and sequential learning (Spiegelhalter and Cowell 1992). We show that exploiting irrelevancies can speedup all those algorithms. Furthermore, it can also speedup the convergence of the EM algorithm.

2 Parameter learning in Bayesian networks

Let x_1, x_2, ..., and x_n be all the variables in a Bayesian network structure and let π_i stand for the set of parents of x_i. This paper assumes all variables have a finite number of possible values. The network structure is a model about the relationships among the variables; it states that the joint probability $p(x_1, x_2, \ldots, x_n)$ can be factorized into the multiplication of the conditional probabilities of the variables given their respective parents, i.e.

$$p(x_1, x_2, \ldots, x_n) = \prod_i p(x_i|\pi_i).$$

This paper is concerned with learning conditional probabilities for a given a network structure. We shall use θ_{ijk} to denote $p(x_i{=}j|\pi_i{=}k)$, where j is a value of variable x_i and k is a combination of the values of the parents of x_i. For convenience, we shall say that k is a value of π_i and and call θ_{ijk} a *parameter pertaining to variable* x_i. We shall also use θ to denote the vector of all the parameters θ_{ijk}. The joint probability over all the variables determined by a parameter vector θ will be denoted by $p(x_1, x_2, \ldots, x_n|\theta)$.

The parameter vector θ is to be estimated from a collection $\mathbf{D}$ of independent data cases D_1, D_2, ..., D_m. Each data case is a set of variable-value pairs. Different data cases might involve different variables. This implies that data cases might be *incomplete* in the sense that they might not contain values for all variables.

If θ is the true parameter vector, then the probability of observing $\mathbf{D}$ is $p(\mathbf{D}|\theta) = \prod_l p(D_l|\theta)$. Viewed as a function of θ, $p(\mathbf{D}|\theta)$ is called the *likelihood function* of θ given data $\mathbf{D}$. The *maximum likelihood (ML) estimate* of θ is a value of θ that maximizes the likelihood function.

When there is a prior probability (density function) $p(\theta)$ about θ, one can compute the posterior probability using Bayes' rule,

$$p(\theta|\mathbf{D}) = cp(\theta)p(\mathbf{D}|\theta),$$

where c is the renormalization constant. In this case, θ_{ijk} can be estimated by using the *posterior expectation* $\int \theta_{ijk} p(\theta_{ijk}|\mathbf{D}) d\theta_{ijk}$.

Lauritzen (1995) and Russell (*et al* 1995) have respectively explained how the EM algorithm (Dempster *et al 1977)* and gradient descent can be used to compute an approximate ML estimate of θ, while Spiegelhalter and

Lauritzen (1990), Spiegelhalter and Cowell (1992), and Titterington (1976) have described approximate methods for obtaining the posterior expectation of θ_{ijk} for every i, j, and k. Although very different in appearances, those methods share one thing in common. In obtaining the next estimate of θ from the current estimate $\theta^{(t)}$, a major part of the computation is spent in calculating the conditional probability

$$p(x_i, \pi_i | D_l, \theta^{(t)})$$

for every variable x_i and for one or every data case D_l. This conditional probability is used to revise the parameters $\theta_{ijk}^{(t)}$ (for all j and all k) pertaining to x_i.

2.1 Purpose of this paper

For any set Y of variables, the *ancestral set* $an(Y)$ consists of variables in Y and ancestors of those variables, i.e. variables from which there are direct paths to at least one variable Y. It is evident that $x_i \in an(Y)$ implies $\pi_i \subset an(Y)$.

Let Y_l be the set of variables whose values are observed in data case D_l. The purpose of this paper is to show

> **Main result**: In various parameter learning algorithms, there is, for any data case D_l, no need to calculate $p(x_i, \pi_i | D_l, \theta^{(t)})$ for variables x_i outside the ancestral set $an(Y_l)$.

The correctness of this result is due to the following proposition.

Proposition 1 *For any data case D_l, the conditional probabilities of variables outside $an(Y_l)$ are irrelevant to $p(D_l|\theta)$.*

Proof: The proposition follows immediately from the fact that

$$p(an(Y_l)|\theta) = \prod_{i \text{ s.t. } x_i \in an(Y_l)} p(x_i | \pi_i, \theta), \tag{1}$$

where for any value j of x_i and any value k of π_i, $p(x_i{=}j | \pi_i{=}k, \theta) = \theta_{ijk}$. □

Intuitively, this proposition implies that a data case D_l should not contribute anything in estimating the parameters θ_{ijk} pertaining to variables x_i

outside $an(Y_l)$, and consequently $p(x_i, \pi_i|D_l, \theta^{(t)})$ needs not be computed for those variables. To formally establish this result for particular algorithms, we will be needing the following lemma.

Lemma 1 *For any* $x_i \notin an(Y_l)$,

$$p(x_i, \pi_i|D_l, \theta) = p(x_i|\pi_i, \theta)p(\pi_i|D_l, \theta).$$

Proof: The lemma follows from the fact that

$$p(an(\{x_i\} \cup Y_l)|\theta) = p(x_i|\pi_i, \theta)p(an(\pi_i \cup Y_l)|\theta).$$

□

The main result is interesting because it cuts down computation and hence speeds-up learning. The amount of computation is reduced not only because the conditional probability $p(x_i, \pi_i|D_l, \theta^{(t)})$ is calculated only for variables x_i in the ancestral set $an(Y_l)$, but also because calculating $p(x_i, \pi_i|D_l, \theta^{(t)})$ for variables x_i outside the ancestral set can be much more difficult than for those in the ancestral set. If one uses the clique tree propagation inference algorithm (Lauritzen and Spiegelhalter 1988, Jensen *et al* 1990), the main result ensures that it suffices to construct a clique tree for the restriction of the Bayesian network onto $an(Y_l)$. Without the main result, one would need to construct a clique tree for the entire network. The maximum clique in the latter clique tree can be much larger than its counterpart in the former clique tree and hence inference can take much more time. It not rare that inference in the entire network is infeasible, while inference in a subnetwork can be done in real time.

The rest of the paper establishes the main result for the three types of algorithms mentioned earlier.

3 The EM algorithm

The EM algorithm (Dempster *et al* 1977) is a general iterative algorithm for computing approximate ML estimates from incomplete data. Lauritzen (1995) has shown how it can be used for parameter learning in Bayesian networks. In this context, major computation at each iteration is spent on calculating the conditional probability $p(x_i, \pi_i|D_l, \theta^{(t)})$ for all data cases

D_l and all variables x_i. This section describes a modification of the EM algorithm (for Bayesian networks). For each data case D_l, the modified algorithm calculates $p(x_i, \pi_i|D_l, \theta^{(t)})$ only for variables x_i in the ancestral set $an(Y_l)$ and hence runs faster than the original algorithm. Moreover, the modified algorithm has a higher rate of convergence than the original algorithm.

3.1 A brief review of the EM algorithm

Let us begin with a brief review of the EM algorithm. The *loglikelihood function* of θ given data $\mathbf{D}$ is defined by

$$l(\theta|\mathbf{D}) = lnp(\mathbf{D}|\theta) = \sum_l lnp(D_l|\theta).$$

It is evident that maximizing the likelihood function is the same as maximizing the loglikelihood function. It is usually more convenient to work with the latter.

A data case D_l is *complete* if all variables are observed in the case. When all the data cases are complete, the loglikelihood function is

$$l(\theta|\mathbf{D}) = \sum_{i,j,k} f(x_i{=}j, \pi_i{=}k) ln\theta_{ijk}, \tag{2}$$

where $f(x_i{=}j, \pi_i{=}k)$ stands for the number of data cases where $x_i{=}j$ and $\pi_i{=}k$. Consequently, the ML estimate θ^* of θ can be obtained by setting

$$\theta^*_{ijk} = \frac{f(x_i{=}j, \pi_i{=}k)}{\sum_j f(x_i{=}j, \pi_i{=}k)} \text{ for all } i,\ j, \text{ and } k. \tag{3}$$

Often, data cases are incomplete. This is when the EM algorithm comes into play. The algorithm starts with an initial estimate $\theta^{(0)}$ of θ and improves it iteratively. From the current estimate $\theta^{(t)}$, the next estimate $\theta^{(t+1)}$ is obtained in two steps: the *expectation step (E-step)* and the *maximization step (M-step)*. The E-step computes the current expected loglikelihood function of θ given data $\mathbf{D}$, i.e.

$$Q(\theta|\theta^{(t)}) = \sum_l \sum_{X_l} lnp(D_l, X_l|\theta)p(X_l|D_l, \theta^{(t)}), \tag{4}$$

where X_l is the set of variables whose values are missing from data case D_l. The M-step chooses the next estimate $\theta^{(t+1)}$ by maximizing the current expected loglikelihood:

$$Q(\theta^{(t+1)}|\theta^{(t)}) \geq Q(\theta|\theta^{(t)}) \text{ for all } \theta.$$

It follows from equation (2) that

$$Q(\theta|\theta^{(t)}) = \sum_{i,j,k} f_t(x_i{=}j, \pi_i{=}k) ln\theta_{ijk},$$

where

$$f_t(x_i, \pi_i) = \sum_l p(x_i, \pi_i|D_l, \theta^{(t)}). \quad (5)$$

Hence, the E-step reduces to computing the function $f_t(x_i, \pi_i)$ for each variable x_i, while the M-step reduces to computing the next estimate of θ by setting,

$$\theta_{ijk}^{(t+1)} = \frac{f_t(x_i{=}j, \pi_i{=}k)}{\sum_j f_t(x_i{=}j, \pi_i{=}k)} \text{ for all } i,\ j, \text{ and } k. \quad (6)$$

From equation (5), we see that at each iteration, the EM algorithm needs to calculate the conditional probability $p(x_i, \pi_i|D_l, \theta^{(t)})$ for every data case D_l and every variable x_i, and this is the dominant part of the computation.

3.2 Modifying the EM algorithm

We say a data case D_l is *upward complete* if $an(Y_l) = Y_l$. By equation (1), we have that when all the data cases are upward complete, the loglikelihood function is

$$l(\theta|\mathbf{D}) = \sum_{i,j,k} g(x_i{=}j, \pi_i{=}k) ln\theta_{ijk}, \quad (7)$$

where $g(x_i{=}j, \pi_i{=}k)$ stands for the number of data cases D_l such that Y_l contains x_i and that $x_i{=}j$ and $\pi_i{=}k$. Consequently, the ML estimate of θ can be obtained by using equation (3) with the f-function replaced by the g-function.

The rest of this subsection describes an EM style algorithm for situations where data cases are not upward complete. An intuitive way to understand the EM algorithm is that it makes an incomplete data case D_l complete by filling in missing data X_l. The algorithm to be described below makes a data case that is not upward complete upward complete by filling in missing data.

Let $\theta^{(t)}$ be the current estimate of θ. We obtain the next estimate in the following two steps. First, we compute the function

$$Q'(\theta|\theta^{(t)}) \quad = \quad \sum_l \sum_{Z_l} lnp_\theta(D_l, Z_l) p(Z_l|D_l, \theta^{(t)}), \tag{8}$$

where Z_l is the set of variables in the ancestral set $an(Y_l)$ whose values are missing from the data case D_l, i.e. $Z_l = an(Y_l)\backslash Y_l$. Then, we choose the next estimate $\theta^{(t+1)}$ by maximizing the Q' function.

It follows from equation (7) that

$$Q'(\theta|\theta^{(t)}) = \sum_{i,j,k} g_t(x_i, \pi_i) ln\theta_{ijk},$$

where

$$g_t(x_i, \pi_i) = \sum_{l \text{ s.t.} x_i \in an(Y_l)} p(x_i, \pi_i|D_l, \theta^{(t)}). \tag{9}$$

Consequently, our first step reduces to computing $g_t(x_i, \pi_i)$ for all variables x_i, and the our second step reduces to obtaining the next estimate $\theta^{(t+1)}$ by setting,

$$\theta_{ijk}^{(t+1)} = \frac{g_t(x_i{=}j, \pi_i{=}k)}{\sum_j g_t(x_i{=}j, \pi_i{=}k)} \text{ for all } i,\ j, \text{ and } k. \tag{10}$$

3.3 Discussions

It is evident that the modified algorithm runs faster than the original EM algorithm. As a matter of fact, in each iteration and for each data case D_l, the former calculates the conditional probability $p(x_i, \pi_i|D_l, \theta^{(t)})$ only for those variables x_i in the ancestral set $an(Y_l)$, while the latter calculates it for all variables.

Because of equation (7), the distribution $p(\mathbf{D}|\theta)$ of upward complete data is from the regular exponential family with natural parameters $ln\theta_{ijk}$. This fact enables one to prove the convergence of the modified algorithm by following the same line of reasoning as in the proof of the convergence of the original EM algorithm (Dempster *et al* 199, Wu 1983).

In the original EM algorithm, the rate of convergence is closely related to the amount of information that is needed to make data cases complete: The greater this amount of information, the slower the rate of convergence. Similarly, the rate of convergence of the modified algorithm is closely related to the amount of information that is needed to make data cases upward complete: The greater this amount of information, the slower the rate of convergence.

Since it take less information to make data cases upward complete than to make them complete, the modified EM algorithm has a higher rate of convergence than the original EM algorithm.

4 Gradient Descent

Finding an ML estimate of θ is a constrainted nonlinear optimization problem. Each parameter θ_{ijk} must be between 0 and 1 and for fixed i and k, $\sum_j \theta_{ijk}$ must be 1.

The gradient projection method (Leunberger 1984) can be used to solve this problem. In this method, the next estimate of θ is obtained from the current estimate $\theta^{(t)}$ as follows: Find the gradient vector $\bigtriangledown_\theta$ of $l(\theta|\mathbf{D})$ at $\theta^{(t)}$; project it onto the constraint surface to get the gradient vector $\bigtriangledown'_\theta$ along the the constraint surface; and take a step in the direction $\bigtriangledown'_\theta$, with the step-size determined by line minization (or rather maximization).

The gradient vector $\bigtriangledown_\theta$ contains a component $\bigtriangledown_{\theta_{ijk}}$ for each parameter θ_{ijk}, which is the partial derivative of $l(\theta|\mathbf{D})$ with respect to θ_{ijk} at $\theta^{(t)}$. Russell *et al* (1995) has shown that

$$\bigtriangledown_{\theta_{ijk}} = \frac{\partial l(\theta|\mathbf{D})}{\partial \theta_{ijk}}|_{\theta=\theta^{(t)}} = \frac{f_t(x_i{=}j, \pi_i{=}k)}{\theta_{ijk}^{(t)}}. \tag{11}$$

The projection $\bigtriangledown'_\theta$ of $\bigtriangledown_\theta$ onto the constraint surface is

$$\bigtriangledown'_{\theta_{ijk}} = \frac{f_t(x_i{=}j, \pi_i{=}k)}{\theta_{ijk}^{(t)}} - \frac{1}{n_i} \sum_j \frac{f_t(x_i{=}j, \pi_i{=}k)}{\theta_{ijk}^{(t)}}, \tag{12}$$

where n_i stands for the number of possible values of x_i.

In each iteration, the conditional probability $p(x_i, \pi_i | D_l, \theta^{(t)})$ needs to be computed for for every data case D_l and for every variable x_i, and this is a major part of the computation. The following proposition states that for each data case D_l, $p(x_i, \pi_i | D_l, \theta^{(t)})$ needs to be computed only for variables x_i in the ancestral set $an(Y_l)$.

Proposition 2 *The quantity one the right hand side of equation (12) remains unchanged if the function f_t is replaced by function g_t.*

Proof: By Lemma 1, we have that for any data case D_l and any $x_i \notin an(Y_l)$,

$$
\begin{aligned}
&\frac{p(x_i{=}j, \pi_i{=}k | D_l, \theta^{(t)})}{\theta^{(t)}_{ijk}} - \frac{1}{n_i} \sum_j \frac{p(x_i{=}j, \pi_i{=}k | D_l, \theta^{(t)})}{\theta^{(t)}_{ijk}} \\
= \; & \frac{p(x_i{=}j | \pi_i{=}k, \theta^{(t)})}{\theta^{(t)}_{ijk}} p(\pi_i{=}k | D_l, \theta^{(t)}) - \frac{1}{n_i} \sum_j \frac{p(x_i{=}j | \pi_i{=}k, \theta^{(t)})}{\theta^{(t)}_{ijk}} p(\pi_i{=}k | D_l, \theta^{(t)}) \\
= \; & p(\pi_i{=}k | D_l, \theta^{(t)}) - \frac{1}{n_i} \sum_j p(\pi_i{=}k | D_l, \theta^{(t)}) \\
= \; & 0.
\end{aligned}
$$

Hence, the proposition follows. □

Russell *et al* (1995) uses an adaptation of the conjugate gradient algorithm, instead of the gradient projection method. There is not enough information in the paper for us to determine whether replacing f_t with g_t there would affect the algorithm behavior in terms of convergence; and if yes, whether the effect is positive or negative.

5 Sequential learning

Sequential learning methods (Spiegelhalter and Lauritzen 1990, Spiegelhalter and Cowell 1992, and Titterington 1976) approximate the posterior expectation $\int \theta_{ijk} p(\theta_{ijk} | \mathbf{D}) d\theta_{ijk}$ for every i, j, and k. They absorb data cases one by one. Suppose $\theta^{(l)}$ is the estimate after absorbing the first l data cases. In absorbing data case D_{l+1}, they compute the conditional probability $p(x_i, \pi_i | D_{l+1}, \theta^{(l)})$ for every variable x_i. This section shows that the conditional probability needs to be computed only for variables in the ancestral set $an(Y_{l+1})$.

To understand the sequential learning methods, let us inductively define

$$p_l(\theta) = cp(D_l|\theta)p_{l-1}(\theta), l = 0, 1, \ldots, m,$$

where c is the renomalization constant and when l=0, $p_0(\theta)$ is simply the prior probability $p(\theta)$. Then $p_m(\theta) = p(\theta|\mathbf{D})$.

Due to the large number of parameter encapsulated in the parameter vector θ, straightforward computation of p_m is unrealistic. Simplifying assumptions, and hence approximations, are usually made.

Let θ_i denote the collection of all the parameters pertaining to variable x_i, i.e. θ_i=$\{\theta_{ijk},$ for all j and $k\}$. The *global independence assumption* states that the parameter groups θ_i for different i's are independent under probability p_l, i.e. $p_l(\theta) = \prod_i p_l(\theta_i)$. When it is not the case, $p_l(\theta)$ is approximated by $\prod_i p_l(\theta_i)$.

For any i and k, define θ_{ik} to be the collection of parameters $\{\theta_{ijk},$ for all $j\}$. The *local independence assumption* states that for any fixed i, the parameter groups θ_{ik} for different k's are independent under probability p_l, i.e. $p_l(\theta_i) = \prod_k p_l(\theta_{ik})$. When it is not the case, $p_l(\theta_i)$ is approximated by $\prod_k p_l(\theta_{ik})$.

The third assumption, which might be called the *Dirichlet distribution assumption*, states that the marginal probability $p_l(\theta_{ik})$ is of a Dirichlet distribution $\mathcal{D}[\alpha^l_{i1k}, \ldots, \alpha^l_{in_ik}]$, where n_i stands for the number of possible values of x_i. When it is not the case, $p_l(\theta_{ik})$ is approximated by a Dirichlet distribution.

The parameters α^l_{ijk} $(j = 1, \ldots, n_i)$ in the Dirichlet distribution can be thought of as representing counts of past cases where x_i=j and π_i=k. The estimate $\theta^{(l)}$ can be obtained from the counts by

$$\theta^{(l)}_{ijk} = \frac{\alpha^l_{ijk}}{\alpha^l_{ik}}, \tag{13}$$

where α^l_{ik}=$\sum_j \alpha^l_{ijk}$ is called the "precision" underlying our belief about $\theta^{(l)}_{ijk}$.

Under those three assumptions, the probability $p_l(\theta)$ is completely determined by the counts α^l_{ijk}. Hence, computing $p_{l+1}(\theta)$ from $p_l(\theta)$ reduces to obtaining the counts α^{l+1}_{ijk} from the counts α^l_{ijk}. We shall refer to this task as count updating.

Spiegelhalter and Lauritzen (1990) have shown that

$$\begin{aligned} p_{l+1}(\theta_{ik}) &= \sum_j p_l(\theta_{ik}|x_i{=}j,\pi_i{=}k)p(x_i{=}j,\pi_i{=}k|D_{l+1},\theta^{(l)}) \\ &+ p_l(\theta_{ik})(1{-}p(\pi_i{=}k|D_{l+1},\theta^{(l)})). \end{aligned} \tag{14}$$

where, by the standard conjugate Bayesian updating, $p_l(\theta_{ik}|x_i{=}j,\pi_i{=}k)$ is of the Dirichlet distribution

$$\mathcal{D}[\alpha^l_{i1k},\ldots,\alpha^l_{ijk}{+}1,\ldots,\alpha^l_{in_ik}].$$

Equation (14) reduces count updating to approximating the distribution $p_{l+1}(\theta_{ik})$ by a Dirichlet distribution. This can be done in several ways. In fractional updating method (Titterington 1976, Smith and Markov 1978), $p_{l+1}(\theta_{ik})$ is approximated by the Dirichlet distribution with counts

$$\alpha^{l+1}_{ijk} = \alpha^l_{ijk} + p(x_i{=}j,\pi_i{=}k|D_{l+1},\theta^{(l)}) \qquad \text{for all } j.$$

The probabilistic editor method (Spiegelhalter and Cowell 1992) approximates $p_{l+1}(\theta_{ik})$ by the Dirichlet distribution with counts

$$\alpha^{l+1}_{ijk} = \alpha^{l+1}_{ik} m_{ijk} \qquad \text{for all } j, \tag{15}$$

where

$$\begin{aligned} m_{ijk} &= \frac{\alpha^l_{ijk}+p^l_{ijk}}{\alpha^l_{ik}+1}, \\ \alpha^{l+1}_{ik} &= \frac{(\alpha^l_{ik}+1)^2\sum_j m^2_{ijk}(1-m_{ijk}) - \sum_j m_{ijk}p^l_{ijk}(1-p^l_{ijk})}{(\alpha^l_{ik}+1)\sum_j m^2_{ijk}(1-m_{ijk}) + \sum_j m_{ijk}p^l_{ijk}(1-p^l_{ijk})}, \end{aligned}$$

and where p^l_{ijk} is a shorthand for $p(x_i{=}j,\pi_i{=}k|D_{l+1},\theta^{(l)})/p(\pi_i{=}k|D_{l+1},\theta^{(l)})$.

A common theme in those methods is that the conditional probability $p(x_i{=}j,\pi_i{=}k|D_{l+1},\theta^{(l)})$ is computed for very variable x_i and all the counts are updated. The following proposition states that one needs only update the counts pertaining to variables in the ancestral set $an(Y_{l+1})$, and consequently $p(x_i{=}j,\pi_i{=}k|D_{l+1},\theta^{(l)})$ needs to be computed only for those variables.

Proposition 3 *For any* $x_i{\notin}an(Y_{l+1})$*,*

$$p_{l+1}(\theta_{ik}) = p_l(\theta_{ik}).$$

Proof: By inspecting the pdf of a Dirichlet distribution, one can easily see that

$$p_l(\theta_{ik}|x_i{=}j, \pi_i{=}k) = p_l(\theta_{ik})\frac{\alpha_{ik}^l}{\alpha_{ijk}^l}\theta_{ijk}.$$

By Lemma 1, we get

$$\begin{aligned} p(x_i{=}j, \pi_i{=}k|D_{l+1}, \theta^{(l)}) &= p(x_i{=}j|\pi_i{=}k, \theta^{(l)})p(\pi_i{=}k|D_{l+1}, \theta^{(l)}) \\ &= \frac{\alpha_{ijk}^l}{\alpha_{ik}^l}p(\pi_i{=}k|D_{l+1}, \theta^{(l)}). \end{aligned}$$

Thus, the proposition follows from the fact that $\sum_j \theta_{ijk}{=}1$. $\Box$.

We would like to point out that the proposition remains true even without the local independence and the Dirichlet distribution assumptions.

6 Conclusions

It is possible to learn parameters for a Bayesian network structure from data because those parameters influence the probability of observing the data. For a particular data case, one can determine, from the given network structure, a subset of parameters that are irrelevant to the probability of observing the data case. Consequently, the data case should not contribute anything in learning those parameters. This paper has shown how such irrelevancies can be exploited to speedup various algorithms for parameter learning in Bayesian networks.

As a future direction, it would be interesting to determine whether irrelevance has a role to play in structure learning.

Acknowledgement

This paper has benefited from the AI and Statistics group at Hong Kong University of Science and Technology. Research was supported by RGC grant HKUST658/95E.

References

[1] A. P. Dempster, N. M. Laird, and D. B. Rubin (1977), Maximum likelihood from incomplete data via the EM algorithm, *Journal of the Royal Statistical Society*, 39 (Series B), 1-38.

[2] D. Heckerman (1995), A tutorial on learning Bayesian networks, Technical Report MSR-TR-95-06, Microsoft Research, Microsoft Corporation.

[3] F. V. Jensen, K. G. Olesen, and K. Anderson (1990), An algebra of Bayesian belief universes for knowledge-based systems, *Networks*, 20, pp. 637 - 659.

[4] M. I. Jordan (1995), Why the logistic function? A tutorial discussion on probabilities and neural networks, Technical Report 9503, Computational Cognitive Science, MIT.

[5] S. L. Lauritzen (1995), The EM algorithm for graphical association models with missing data, *Computational Statistics and Data Analysis*, Vol. 19, 191-201.

[6] S. L. Lauritzen and D. J. Spiegelhalter (1988), Local computations with probabilities on graphical structures and their applications to expert systems, *Journal of Royal Statistical Society B*, 50: 2, pp. 157 - 224.

[7] D. G. Leunberger (1984), *Linear and Nonlinear Programming*, Second edition , Addison-Wesley Publishing Company.

[8] J. Pearl (1988), *Probabilistic Reasoning in Intelligence Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers, Los Altos, CA.

[9] S. J. Russell, J. Binder, D. Koller, and K. Kanazawa (1995), Local learning in probabilistic networks with hidden variables, *IJCAI'95*, 1146-1152.

[10] D. J. Spiegelhalter and R. G. Cowell (1993), Learning in probabilistic expert systems, in *Bayesian Statistics 4*, J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith (eds.), 447-465.

[11] D. J. Spiegelhalter, A. P. Dawid, S. L. Lauritzen, and R. G. Cowell (1993), Bayesian analysis in expert systems, *Statistical Science*, Vol. 8, No. 3, 219-283.

[12] C. F. J. Wu (1983), On the convergence properties of the EM algorithm, *The Annals of Statistics*, Vol. 11, No. 1, 95-103.