Removing Redundancy in the Expectation-Maximization Algorithm for Bayesian Network Learning

FABIO GAGLIARDI COZMAN*

Escola Politécnica, University of São Paulo
Av. Prof. Mello Moraes, 2231, 05508-900, São Paulo, SP - Brazil
fgcozman@usp.br

Abstract. This paper focuses on parameter estimation in Bayesian networks with fixed structure. The paper develops techniques that avoid redundant computation in the Expectation-Maximization algorithm. The resulting algorithm is analyzed in the context of databases with relatively few missing elements. The analysis of the algorithm is based on the expected of requisite variables in Bayesian network inference, a quantity of theoretical and practical interest in the theory of Bayesian networks.

1 Introduction

Bayesian networks represent statistical data using directed acyclic graphs. A Bayesian network is usually built using either expert knowledge about a domain, or data collected on the domain, or both. This paper looks at the situation where the structure of the domain has already been translated into a directed acyclic graph. The problem is to estimate the probability values associated with the graph, an activity called *parameter estimation* or *parameter learning*. The focus here is on parameter estimation with missing data, where a small (but not insignificant) portion of a database is randomly missing. Such databases are often handled by the Expectation-Maximization algorithm in Bayesian network learning [5].

The paper develops techniques that avoid redundant computation in the EM algorithm. The main idea is to detect the requisite variables for any query in the EM algorithm and to eliminate identical sets of requisite variables. The performance of the resulting algorithm is dependent on the expected number of requisite variables in Bayesian network inference, a quantity of theoretical and practical interest. An approximate model for this quantity is presented in this paper, serving as a first analysis of requisite variables in Bayesian network inference.

Section 2 reviews the basic concepts in parameter estimation for Bayesian networks. Section 3 investigates parameter estimation with missing data and presents the main algorithm. Section 4 develops a theoretical model for the expected number of requisite variables in Bayesian network inference.

*The author was partially supported by CNPq through grant 300183/98-4.

2 Estimating parameters in Bayesian networks

Bayesian networks are graphical representations for statistical knowledge. In a Bayesian network, a joint probability density over a set of variables $\mathbf{X}$ is specified by a directed acyclic graph. Each node of this graph represents a variable X_i and each variable is associated with a conditional probability density $p(X_i|\text{pa}(X_i))$, where $\text{pa}(X_i)$ denotes the parents of X_i in the graph. The basic assumption adopted in Bayesian networks is that every variable is independent of its nondescendants nonparents by its parents. This assumption implies [9]:

$$p(\mathbf{X}) = \prod_i p(X_i|\text{pa}(X_i)) \,. \tag{1}$$

The *Markov blanket* of X_i is the set of variables containing (a) the parents of X_i; (b) the children of X_i; (c) the parents of children of X_i. Expression 1 implies that a variable X_i is independent of every other variable in the network given X_i's Markov blanket.

Given a Bayesian network, it is often necessary to compute the posterior probability of query variables $\mathbf{X}_q$ conditional on a set E of observations (the *evidence*). The computation of the density $p(\mathbf{X}_q|E)$ is called the *inference process*. If the value of X_i affects the computation of $p(\mathbf{X}_q|E)$, then X_i is a *requisite* variable for $p(\mathbf{X}_q|E)$. There are polynomial algorithms that detect requisite variables using d-separation [4, 10] or moral graph separation [6, 8]. Given a Bayesian network, the *moral graph* of the network is the undirected graph obtained by connecting parents of each variable and then removing the direction of the arrows in the network. To determine the variables that are d-separated from $\mathbf{X}_q$ by E, it is necessary to: (a) select the sub-graph containing $\mathbf{X}_q$, the observed variables, and the ancestors of these variables; (b) create the moral graph of the selected sub-graph; (c) use standard graph separation with respect to E. If we gather the connected variables (in the moral graph) and add the observed variables that block paths from $\mathbf{X}_q$, we construct a set containing the requisite variables for $p(\mathbf{X}_q|E)$.

In this paper we deal with Bayesian network estimation from data. Assume that the struture of a network has been specified: the directed acyclic graph in the network has been constructed by experts or by data analysis. The parameters of a Bayesian network are probability values. Denote by θ_{ijk} the probability value $p(X_i = x_{ij}|\text{pa}(X_i) = \pi_{ik})$, where x_{ij} is a value of X_i and π_{ik} are values for the parents of X_i. The estimates are produced using data in a database. The database is organized as a table, where each row is called a *record* — a round of measurements for all variables in the network. Records are denoted by D_m; the whole database is indicated by $\mathbf{D}$. If a variable is not instantiated in a particular record, that variable appears as *missing data*.

Two popular estimation methods are maximum likelihood and maximum a posteriori estimation [5]. Denote by Θ a vector containing all parameters θ_{ijk}. A maximum likelihood estimate is a value $\hat{\Theta}$ that maximizes the likelihood function $p(\mathbf{D}|\Theta)$. Note that as $\mathbf{D}$ is fixed, the likelihood function is a function only of Θ. Maximum likelihood estimates can also be obtained by maximizing the log-likelihood function, $\log p(\mathbf{D}|\Theta)$. If all records are independent conditional on the parameters, we can decompose the log-likelihood:

$$\hat{\Theta} = \arg\max_{\Theta} \sum_m \log p(D_m|\Theta) .$$

In a Bayesian network, the log-likelihood function $p(D_m|\Theta)$ is a multivariate multinomial distribution.

A maximum a posteriori is an estimate $\hat{\Theta}$ that maximizes the posterior distribution $p(\Theta|\mathbf{D})$:

$$\hat{\Theta} = \arg\max_{\Theta} p(\Theta|\mathbf{D}) = \arg\max_{\Theta}(p(\mathbf{D}|\Theta)\, p(\Theta)\, /p(\mathbf{D})).$$

A maximum a posteriori estimate depends on the prior distribution $p(\Theta)$. This prior distribution is usually taken to be a Dirichlet distribution with independence relations among parameters [5].

As an example, suppose the database $\mathbf{D}$ does not contain missing data; every record is a complete instantiation of variables. The maximum likelihood estimate for θ_{ijk} is (N_{ijk}/N_{ik}), where N_{ijk} is the number of records where the event $\{X_i = x_{ij}\} \cap \{\text{pa}(X_i) = \pi_{ik}\}$ obtains, and N_{ik} is the number of records where the event $\{\text{pa}(X_i) = \pi_{ik}\}$ obtains. Similar expressions are obtained for maximum a posteriori estimation with Dirichlet priors and appropriate independence assumptions [5]. Estimation is then reduced to counting. Note that if two records are identical, the same counters N_{ijk} and N_{ik} are updated twice. To avoid redundant counting, we could sort the records and ignore duplicates. For complete data, the effort spent in sorting records is bound to surpass the effort spent in counting. But the removal of redundant information turns out to be important in the presence of missing data, as discussed in the next section.

3 Missing data: improving EM

Suppose now that we have a database with missing data. It is often the case that a database has a significant amount of missing data, but still the great majority of entries are not missing.

The existence of missing data complicates the whole estimation process, because there are no closed-form expression for the estimates. Numerical methods must be used to maximize likelihood, posterior densities or other objective functions. The Expectation-Maximization (EM) algorithm is an iterative scheme that produces maximum likelihood estimates. When applied to Bayesian networks, the EM algorithm produces a simple updating scheme: At iteration n, the estimate $\hat{\theta}^n_{ijk}$ is

$$\frac{\sum_m p\Big(X_i = x_{ij}, \text{pa}(X_i) = \pi_{ik}|D_m, \hat{\Theta}^{n-1}\Big)}{\sum_m p\Big(\text{pa}(X_i) = \pi_{ik}|D_m, \hat{\Theta}^{n-1}\Big)}. \quad (2)$$

The densities are written as conditional on $\hat{\Theta}^{n-1}$, meaning that computations for $\hat{\Theta}^n$ must be performed with the parameters $\hat{\Theta}^{n-1}$.

For each record in a database, the EM algorithm requests an inference run. Now, if two records are identical, one of them should be removed: it is only necessary to add inference results twice in the appropriate summations. This idea is not new:

> For computational efficiency one would of course collect cases with identical observations in groups and only calculate $p(i_a|i_b)$ *[the probability densities required in the EM algorithm]* once for each of these identical terms [7].

The idea of grouping records has also been used by Zhang [11] in the context of high percentages of missing data; his algorithm is quite similar to the one presented in this paper. But here the focus is on databases with low percentages of missing data.

Despite the clear advantage of looking for duplicate records, the probability that two complete records are identical is likely to be quite small for networks of medium/high size. To improve the probability of detecting redundancies, we must compare sub-sets of database columns instead of complete records.

The main idea is: Instead of comparing complete records, we should compare the requisite variables that are effectively used in the EM algorithm. The resulting algorithm can be described as follows.

1. Start with some initial guess Θ^0 and repeat for n larger than zero:
2. Mark every element of the database by the flag NOT_PROCESSED.
3. For each variable X_i:
   (a) Create an empty binary tree.
   (b) For each variable X_i, go over the records. If X_i is marked as PROCESSED in record D_m, skip D_m.
   (c) If X_i is NOT_PROCESSED in D_m, detect the requisite variables for the computation of the density $p\left(X_i, \text{pa}(X_i)|D_m, \hat{\Theta}^{n-1}\right)$. Denote these variables and their instantiations by R_{im}.
      - If R_{im} is not in the binary tree:
        i. Add R_{im} to the binary tree;
        ii. Associate a counter with R_{im} and initialize the counter with zero.
        iii. Mark every variable in R_{im} as PROCESSED.
      - If R_{im} is in the binary tree:
        i. Increment by one the counter associated with R_{im};
        ii. Ignore the record D_m for variable X_i.
   (d) For every leaf in the binary tree, take the stored set of variables R_{im}. Then compute the density $p\left(X_i, \text{pa}(X_i)|D_m, \hat{\Theta}^{n-1}\right)$ for every variable in R_{im} (this can be done in two passes through the variables, using generalized variable elimination [3] or junction tree methods [6]). The binary tree contains counters that indicate how many times to repeat summations in Expression (2)

The algorithm constructs binary trees to store requisite variables and to detect duplicate requisite variables. The binary tree can order R_{im} lexicographically, sorting first with respect to the number of requisite variables and then with respect to the values of requisite variables. Records without missing values will then be processed before records with missing values.

The flags PROCESSED and NOT_PROCESSED are used because inference algorithms can compute posterior marginal densities for all requisite variables in a query (as enforced by step 3.d).

Note that the various sets of requisite variables are "almost" disjoint (sets are disjoint except for the possibility of shared observed variables). Consequently, the total number of variables stored in the leaves of all binary trees is approximately equal to the number of elements in the database of records. Because a binary tree uses twice as space as its number of leaves, the total memory consumed by the binary trees for all variables is approximately twice the size of the database of records.

If enough computer memory is available, the binary trees for all variables should be computed once and held in memory. When this is possible, it is better to store the various R_{im} in a more refined form: The variables in R_{im} should be properly ordered for processing by a generalized variable elimination or junction tree algorithm. By doing so, the actual computation to be performed at every iteration of the EM algorithm is reduced to a minimum.

The performance of the algorithm depends on the number of requisite variables for any query $p(X_i, \text{pa}(X_i)|D_m)$. The important question is: In a Bayesian network, what is the expected number of variables that are requisite for a query of the type $p(X_i, \text{pa}(X_i)|D_m)$? Finding an analytic expression for the expected number of requisite variables seems to be a difficult task; here we investigate this expected number somewhat informally.

4 The expected number of requisite variables

Suppose that some percentage P of values in a database are missing; take P to be smaller than 5%. Assume that missing data occur in a random fashion; any variable has probability P of being missed at any time. Note that if P is too small, it may make sense to discard the records with missing data — but for a network with 30 variables and P equal to 0.02, we should expect to discard almost half of the database (as $(1-P)^{30} \cong 0.54$). It seems reasonable to investigate values of P that are larger than 0.01.

We are interested in the number of requisite variables for an arbitary query $p(X_0, \text{pa}(X_0)|D_m)$. Denote by Q_i the number of requisite variables for query $p(X_i, \text{pa}(X_i)|D_m)$. Denote by S_i the number of parents of variable X_i, and denote by B_i the number of variables in the Markov blanket of X_i. Assume the S_i to be independent of each other and to have a Poisson distribution with parameter λ (that is, $p(S_i) = e^{-\lambda}\lambda^{S_i}/S_i!$, and the expected value of S_i is λ). It seems reasonable to focus on small values for λ. Networks with large number of parents, like the CPCS network, seem to be rather uncommon. In fact, Table 1 indicates that the average number of parents in networks in the literature is surprisingly low.

Note that X_i and $\text{pa}(X_i)$ are fully observed in record D_m with probability $(1-P)^{S_i+1}$. The expected percentage of records that contain a complete set of observations for a variable and its parents is:

$$\sum_{S=0}^{\infty}(1-P)^{S+1}\frac{e^{-\lambda}\lambda^S}{S!} =$$

$$(1-P)e^{-\lambda P}\sum_{S=0}^{\infty}\frac{e^{-\lambda(1-P)}(\lambda(1-P))^S}{S!} =$$

Bayesian network	Number of nodes	Total number of parents	Average number of parents per variable
Alarm [2]	37	46	1.2432432
Asia [8]	8	8	1.0
Hailfinder [1]	56	66	1.1785715
Munin (part 1) [6]	189	282	1.4920635

Table 1: Bayesian networks in the literature and their average number of parents per variable.

$$e^{-\lambda P}(1-P).$$

Table 2 shows the expected percentage of complete observations for various values of P and λ. Note that the majority of records are expected not to be used in inference. Even at high values of P (10%) and λ (4) the percentage of complete records exceeds 75%; only for the most extreme combinations of P and λ the percentage of complete records goes below this mark.

When at least one of $\{X_i, \mathrm{pa}(X_i)\}$ is missing, it becomes necessary to perform inference (Expression (2)). We are interested in $E[Q_0]$, the expected number of requisite variables for query $p(X_0, \mathrm{pa}(X_0)|D_m)$. The following approximations are made:

- As we search for requisite variables, we never reach the boundary of the network. The network extends indefinitely in all directions as a regular structure.
- When a new requisite variable is found, the Markov blanket of this variable does not contain any requisite variables previously found.

Both approximations overestimate Q_0. Denote by T_i the number of variables contained in the following set $\mathcal{T}_i$: For a variable X_i, add the Markov blanket to $\mathcal{T}_i$; for any variable in $\mathcal{T}_i$ that is not observed, add the Markov blanket of this variable to $\mathcal{T}_i$ until no variable can be added to $\mathcal{T}_i$.

The probability that X_0 and $\mathrm{pa}(X_0)$ are all observed is $(1-P)^{S_0+1}$. If $\{X_0, \mathrm{pa}(X_0)\}$ are observed, there are (S_0+1) requisite variables. Now, if only X_0 is observed, there are T_0 requisite variables for $p(X_0, \mathrm{pa}(X_0)|D_m)$. If X_0 is observed, then the fact that any of $\mathrm{pa}(X_0)$ is observed does not add new requisite variables. The converse is true: if any of $\mathrm{pa}(X_0)$ is observed, the fact that X_0 is observed or not does not change the number of requisite variables. So, take T_0 to mean the number of variables in the set $\mathcal{T}_0$ constructed from any observed variable in $\{X_0, \mathrm{pa}(X_0)\}$. Consequently there is probability $(1-(1-P)^{S_0+1})$ that the number of requisite variables is T_0. So, Q_0 is bounded by

$$Q_0' = (1-P)^{S_0+1}(S_0+1) + (1-(1-P)^{S_0+1})(T_0+1). \quad (3)$$

The value of a generic T_i is equal or larger than B_i (the size of X_i's Markov blanket). Now, each element of X_i's Markov blanket can be either observed or not. Suppose X_j belongs to X_i's Markov blanket and is not observed. Denote by I_j the indicator function that is 0 if X_j is observed and 1 if X_j is not observed. We obtain a recursion for T_i:

$$T_i = B_i + \sum_{j \neq i} I_j T_j,$$

where the summation goes over the Markov blanket of X_i. Now we can obtain the expected value of T_i:

$$\begin{aligned} E[T_i] &= E[B_i] + E\left[\sum_{j\neq i} I_j T_j\right] \\ &= E[B_i] + E\left[E\left[\sum_{j\neq i} I_j T_j | B_i\right]\right] \\ &= E[B_i] + E[P(B_i-1)E[T]], \end{aligned}$$

where we use the fact that $E[T_i]$ is equal to a common value $E[T]$ for every variable X_i. As an approximation, adopt independence among the various stages of the recursion. Taking the expected value of B_i to be a common value $E[B]$ for every variable X_i,

$$\begin{aligned} E[T_i] &= E[T] = E[B] + pE[T](E[B]-1) \\ \Rightarrow E[T] &= \frac{E[B]}{1-P(E[B]-1)}. \end{aligned} \quad (4)$$

We have:

$$\begin{aligned} E[Q_0'] &= E\left[(1-P)^{S_0+1}(S_0+1)\right] + \\ &\quad E\left[(1-(1-P)^{S_0+1})\times \right. \\ &\quad \left.\left(1 + B_0 + PB_0\frac{E[B]}{1-P(E[B]-1)}\right)\right] \\ &= \left(e^{-\lambda P}(1-P)\right)(1+\lambda(1-P)) + \\ &\quad E\left[(1-(1-P)^{S_0+1})\times \right. \\ &\quad \left.\left(1 + \frac{B_0}{1-P(E[B]-1)}\right)\right]. \end{aligned}$$

The relationship between $E[S_i]$ and $E[B_i]$ appears to be quite complex; in fact, it seems reasonable to assume

	$\lambda=1$	$\lambda=2$	$\lambda=3$	$\lambda=4$
$P=0.01$	0.980149	0.970397	0.960741	0.951182
$P=0.02$	0.960595	0.941574	0.922929	0.904654
$P=0.03$	0.941332	0.913512	0.886513	0.860313
$P=0.04$	0.922358	0.886192	0.851444	0.818058
$P=0.05$	0.903668	0.859596	0.817673	0.777794
$P=0.06$	0.885259	0.833705	0.785154	0.73943
$P=0.07$	0.867126	0.808503	0.753843	0.702879
$P=0.08$	0.849267	0.783972	0.723698	0.668057
$P=0.09$	0.831677	0.760096	0.694675	0.634885
$P=0.10$	0.814354	0.736858	0.666736	0.603288

Table 2: Values for the expected percentage of completely observed $\{X_i, \text{pa}(X_i)\}$, $(e^{-\lambda P}(1-P))$.

	$\lambda=1$	$\lambda=2$	$\lambda=3$	$\lambda=4$
$P=0.01$	2.0422	3.14268	4.29744	5.51327
$P=0.02$	2.08905	3.31382	4.68483	6.24204
$P=0.03$	2.14093	3.51909	5.19331	7.30139
$P=0.04$	2.19829	3.76602	5.87161	8.91319
$P=0.05$	2.26162	4.06469	6.80058	11.5663

Table 3: Approximations for the expected number of requisite variables.

that this relationship is a *property* of practical Bayesian networks, not a quantity that should be derived from assumptions regarding the number of parents in the network. Consider the following heuristic arguments regarding the behavior of $E[B_i]$. It seems reasonable to assume that $E[B_i]$ is smaller than $E[S_i](E[S_i]+1)$, because a variable should have $E[S_i]$ parents and $E[S_i]$ children on average, and each children has $E[S_i]$ parents on average — if we take these values and assume that none of the parents and children are identical, we find some $E[S_i](E[S_i]+1)$ distinct variables. This quadratic relationship seems quite pessimistic, because in practice there are repetitions within the Markov blanket of a variable. At least for small values of S_i, it seems reasonable to use the approximation that $B_0 = \alpha S_0$ and $E[B_i] = \alpha E[S_i]$. We obtain the approximation:

$$\begin{aligned} E[Q_0'] &= \left(e^{-\lambda P}(1-P)\right)(1+\lambda(1-P)) + \\ & \quad E\left[(1-(1-P)^{S_0+1})\left(1+\alpha\frac{S_0}{1-P(\alpha\lambda-1)}\right)\right] \\ &= \left(e^{-\lambda P}(1-P)\right)(1+\lambda(1-P)) + \\ & \quad \left(1-e^{-\lambda P}(1-P)^2\right)\frac{\alpha\lambda}{1-P\alpha\lambda}. \end{aligned}$$

Table 3 shows the approximation for Q_0 under various values of P and λ, with α fixed at 3.

Note that the expected number of requisite variables is consistently low, reaching relatively large values only for $\lambda = 4$. This indicates that the algorithm presented in this paper is actually a promising method: It can reduce computation in the EM algorithm when the percentage of missing data is low.

5 Conclusion

The algorithm presented in this paper represents an initial investigation into methods that remove redundant computations in Bayesian network estimation. The algorithm presented here can be used profitably when a relatively large database contains a low, but not insignificant, percentage of missing data. Other algorithms, like gradient descent optimization, require computations that are similar to the EM algorithm, and can benefit from the techniques presented here.

Future work will concentrate on two fronts. The first, theoretical, is the investigation of accurate bounds for the expected number of requisite variables in Bayesian networks. The second, empirical, is the study of how much computation is saved in practice by the algorithm presented in this paper.

References

[1] B. Abramson, J. Brown, A. A. Murphy, and R. L. Winkler. Hailfinder: A Bayesian system for forecasting severe weather. *Int. Journal of Forecasting*, 12:57–71, 1996.

[2] I. Beinlich, H. J. Suermondt, R. M. Chavez, and G. F. Cooper. The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. *Second European Conf. on Artificial Intelligence in Medicine*, pages 247–256, 1989.

[3] F. G. Cozman. Generalizing variable elimination in Bayesian networks. To appear, *Workshop on Prob-*

abilistic Reasoning in Artificial Intelligence at IBERAMIA, Atibaia, São Paulo, Brazil, 2000.

[4] D. Geiger, T. Verma, and J. Pearl. Identifying independence in Bayesian networks. *Networks*, 20:507–534, 1990.

[5] D. Heckerman. A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, March 1995 (revised November, 1996).

[6] F. V. Jensen. *An Introduction to Bayesian Networks*. Springer Verlag, New York, 1996.

[7] S. L. Lauritzen. The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, pages 191–201, 1995.

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

[9] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, California, 1988.

[10] R. D. Shachter. Bayes-ball: The rational pastime (for determining irrelevance and requisite information in belief networks and influence diagrams). In G. F. Cooper and S. Moral, editors, *XIV Conf. on Uncertainty in Artificial Intelligence*, San Francisco, California, United States, 1998. Morgan Kaufmann.

[11] N. L. Zhang. Irrelevance and parameter learning in Bayesian networks. *Artificial Intelligence*, 88:359–373, 1996.