An Improved Structural EM to Learn Dynamic Bayesian Nets

Cassio P. de Campos
IDSIA
cassio@idsia.ch

Zhi Zeng
RPI
zengz@rpi.edu

Qiang Ji
RPI
jiq@rpi.edu

Abstract

This paper addresses the problem of learning structure and parameters of Bayesian and Dynamic Bayesian networks from data based on the Bayesian Information Criterion. We describe a procedure to map the problem of the dynamic case into a corresponding augmented Bayesian network through the use of structural constraints. Because the algorithm is exact and anytime, it is well suitable for a structural Expectation–Maximization (EM) method where the only source of approximation is due to the EM itself. We show empirically that the use a global maximizer inside the structural EM is computationally feasible and leads to more accurate models.

1 Introduction

A Bayesian network (BN) is a probabilistic graphical model that relies on a structured dependency among random variables to represent a joint probability distribution in a compact and efficient manner. While BNs are unrelated to time sequences, Dynamic Bayesian Networks (DBNs) model temporal processes. Assuming Markovian and stationary properties, DBNs may be encoded in a very compact way and inferences are executed quickly. Learning the structure of these networks from data is one of the most challenging problems, specially when data are incomplete.

The problem is to find the best directed acyclic graph (DAG) according to some score function that depends on the data [5]. Best exact ideas are based on dynamic programming [6], and they spend time and memory proportional to $n \cdot 2^n$, where n is the number of variables, or branch-and-bound (B&B) techniques [2], with better average but poorer worst case. For incomplete data sets, the availability of methods is reduced. The main method is the structural EM (called simply SEM) [3], which has also been investigated in the context of DBNs [4].

This paper aims at learning the structure of BNs and DBNs from incomplete data using the Bayesian Information Criterion (BIC). For the DBN case, we describe a procedure to map its structural learning problem into a corresponding augmented BN through the use of constraints, so that the same exact algorithm can be employed in both contexts. Because the B&B algorithm may be stopped at any moment with an approximate solution, it is well suitable for a SEM without the huge computational expense of other exact methods, while keeping global guarantees, and the only source of approximation is due to the EM method itself. We show empirically that the use a global maximization procedure inside the SEM is computationally attractive and leads to more accurate models.

2 BNs and DBNs

A BN represents a joint probability distribution over a collection of random variables. It can be defined as a triple $(\mathcal{G}, \mathcal{X}, \mathcal{P})$, where $\mathcal{G} = (V_\mathcal{G}, E_\mathcal{G})$ is a DAG with $V_\mathcal{G}$ a collection of n nodes associated to random variables $\mathcal{X}$ (a node per variable), and $E_\mathcal{G}$ a collection of arcs; $\mathcal{P}$ is a collection of conditional probabilities $p(X_i|PA_i)$ where PA_i denotes the parents of X_i in the graph (PA_i may be empty), respecting the relations of $E_\mathcal{G}$. We assume throughout that variables are categorical. In a BN every variable is conditionally independent of its non-descendants given its parents (Markov condition). This structure induces a joint probability distribution by the expression $p(X_1, \ldots, X_n) = \prod_i p(X_i|PA_i)$. Let $r_i \geq 2$ be the number of discrete categories of X_i, q_i the number of instantiations of the parent set, that is, $q_i = \prod_{X_t \in PA_i} r_t$, and θ be the entire vector of parameters such that $\theta_{ijk} = p(x_i^k|pa_i^j)$, where $i \in \{1, \ldots, n\}$, $j \in \{1, ..., q_i\}$, $k \in \{1, ..., r_i\}$.

Given a complete data set $D = \{D_1, \ldots, D_N\}$

with N instances, where $D_u = \{x_{1,u}^{k_1}, \ldots, x_{n,u}^{k_n}\}$ is an instance of all the variables, the goal is to find a graph $\mathcal{G}$ that maximizes BIC:

$$\max_{\mathcal{G}} s_D(\mathcal{G}) = \max_{\theta}(L_D(\theta) - t \cdot \frac{\log N}{2}),$$

where θ and $\mathcal{G}$), $t = \sum_{i=1}^{n}(q_i \cdot (r_i - 1))$, the number of free parameters, depend on the graph, and $L_D(\theta) = \sum_{i=1}^{n} L_{D,i}(\theta_i)$, where $L_{D,i}(\theta_i) = \sum_{j=1}^{q_i}\sum_{k=1}^{r_i} n_{ijk} \log \theta_{ijk}$, is the log-likelihood function. n_{ijk} indicates how many elements of D contain both x_i^k and pa_i^j. From now on, the subscript D is omitted for simplicity. An important property of BIC is that it is decomposable, that is, it can be computed at each node X_i separately: $\max_{\mathcal{G}} s(\mathcal{G}) = \max_{\mathcal{G}} \sum_{i=1}^{n} s_i(PA_i)$, where $s_i(PA_i) = L_i(PA_i) - t_i(PA_i) \cdot \frac{\log N}{2}$, with $L_i(PA_i) = \max_{\theta_i} L_i(\theta_i)$ (θ_i is the parameter vector related to X_i, so it depends on the choice of PA_i), and $t_i(PA_i) = q_i \cdot (r_i - 1)$. Because of this property and to avoid computing such functions several times, we create a cache that contains $s_i(PA_i)$ for each X_i and each parent set PA_i. Note that the cache could have an exponential size on n, as there are 2^{n-1} subsets of $\{X_1, \ldots, X_n\} \setminus \{X_i\}$ to be considered as parent sets, but the practical size of the cache is very small (by using the properties described in [2]).

While BNs are not directly related to time, DBNs are used to model temporal processes. We assume that the time is discrete and that a Markovian property holds. Thus, if $X_{i,t} \in \mathcal{X}_t$ are the variables at time t, we have that $p(\mathcal{X}_{t+1}|\mathcal{X}_0, \ldots \mathcal{X}_t) = p(\mathcal{X}_{t+1}|\mathcal{X}_t)$. Furthermore, the process is assumed to be stationary, that is, $p(\mathcal{X}_{t+1}|\mathcal{X}_t)$ is independent of t. Given such assumptions, DBNs can be viewed as two-slice temporal BNs, where at time zero, we have a standard BN, which we denote $\mathcal{B}^o$, and for slices 1 to T there is another BN (called *transitional* BN and denoted simply $\mathcal{B}$) defined over the same variables but where nodes may have parents on two consecutive slices, that is, $\mathcal{B}$ defines the distribution $p(\mathcal{X}_{t+1}|\mathcal{X}_t)$.

To learn a DBN, we assume that many temporal sequences of data are available. Thus, a complete data set $D = \{D_1, \ldots, D_N\}$ is composed of N sequences, where each D_u is composed of instances $D_{u,t} = \{x_{1,u,t}^{k_1}, \ldots, x_{n,u,t}^{k_n}\}$ for $t = 0, \ldots, T$ (where T is the total number of slices/frames apart from the initial one). Note that there is an implicit order among the elements of each D_u. As conditional probability distributions for time $t > 0$ share the same parameters, we can unroll the DBN to obtain the factorization $p(\mathcal{X}_{1:T}) = \prod_i p^0(X_i|\pi_i)\prod_{t=1}^{T}\prod_i p(X_{i,t}|\pi_{i,t})$, where $p^0(X_i|\pi_i)$ are the local conditional distributions of $\mathcal{B}^0$, $X_{i,t}$ and $\pi_{i,t}$ represent the corresponding variables in time t, and $p(X_{i,t}|\pi_{i,t})$ are the local distributions of $\mathcal{B}$. Learning parameters of DBNs is similar to learning parameters of BNs, but we deal with counts n_{ijk} for both $\mathcal{B}^0$ and $\mathcal{B}$. The counts related to $\mathcal{B}^0$ are obtained from the first slice of each sequence, so there are N samples overall, while counts for $\mathcal{B}$ are obtained from the whole time sequences, so we have $N \cdot T$ elements to consider (each sequence has the same size for ease of expose). The score function of a given structure decomposes between the score function of $\mathcal{B}^0$ and the score function of $\mathcal{B}$, so we look for graphs such that $\max_{\mathcal{G}^0,\mathcal{G}} s(\mathcal{G}^0) + s(\mathcal{G}) =$

$$\max_{\theta^0}(L^0(\theta^0) - t^0 \cdot \frac{\log N}{2}) + \max_{\theta}(L(\theta) - t \cdot \frac{\log(N \cdot T)}{2}), \quad (1)$$

where t^0 and t are respectively the number of free parameters in $\mathcal{B}^0$ and $\mathcal{B}$. Counts are obtained from data separately for the initial and the transitional BNs, and the problem reduces to the learning problem in a BN. To learn the structure of a DBN, the procedure is roughly equivalent. We must take care of learning both the initial and the transitional networks. We develop a transformation of the structure learning problem to a corresponding structure learning problem in an augmented BN.

1. Learn $\mathcal{B}^0$ using the data set $\{D_{1,0}, \ldots, D_{N,0}\}$ of the first slice of each sequence. Note that this is already a standard BN structure learning and thus we obtain the graph $\mathcal{G}^0$ for the first maximization of Equation (1).

2. Create a BN $\mathcal{B}' = (\mathcal{G}', \mathcal{X}', \mathcal{P}')$ with twice as many nodes as $\mathcal{B}$. Denote the nodes as $(X_1, \ldots, X_n, X_1', \ldots, X_n')$. Construct a new data set D' that is composed by $N \cdot T$ elements $\{\forall_t(D_{1,t-1}, D_{1,t}), \ldots, \forall_t(D_{N,t-1}, D_{N,t})\}$ for every time sequence $1 \leq u \leq N$ and every slice $0 < t \leq T$. Note that each $(D_{u,t-1}, D_{u,t})$ is a complete sample for the variables of $\mathcal{B}'$, with elements of two consecutive slices.

3. Include structural constraints as follows:

$$\forall_{1 \leq i \leq n}\ arc(X_i, X_i') \quad (2)$$

$$\forall_{1 \leq i \leq n}\ indegree(X_i, 0, eq) \quad (3)$$

Equation (2) forces the time relation between the same variable in consecutive time slices. This constraint might be discarded if someone does not want each variable to be correlated to itself of the past slice. Equation (3) forces the variables $X_1, \ldots, X_n$ to have no parents. These structural constraints are imposed during the construction of the cache of scores so only valid parent sets are stored.

4. Learn $\mathcal{B}'$ using the data set D' with the BN structure learning procedure. Note that the parent sets of $X_1, \ldots, X_n$ are already fixed to be empty, so the output graph will maximize

$$\max_{\theta'_1,\ldots,\theta'_n} (\sum_i L_i(\theta'_i) - t \cdot \frac{\log(N \cdot T)}{2}),$$

where $L_i(\theta'_i)$ is the log-likelihood of node X'_i and $t = \sum_i t'_i$ is the number of free parameters in the nodes $X'_1, \ldots, X'_n$. This holds because of the decomposability of the score function among nodes, so that the scores of the nodes $X_1, \ldots, X_n$ are fixed and can be removed from the maximization (they are constant).

5. Finally, we take the subgraph of $\mathcal{G}'$ corresponding to the variables $X'_1, \ldots, X'_n$ to be the graph of the transitional BN $\mathcal{B}$. This subgraph has arcs among $X'_1, \ldots, X'_n$ (which are arcs correlating variables of the same time slice) as well as arcs from the previous slice to them.

Therefore, after applying this transformation, the structure learning problem in a DBN roughly becomes two calls to the structure learning in a BN.

3 Incomplete Data

We assume in this paper that all variables (even hidden ones) have known domains. The most common situation is to assume that values are missing at random, so that we marginalize to obtain a function of the observed variables. However, this function becomes hard to evaluate, and other problems arise: for instance, the ideas to build a cache of local scores cannot be applied anymore. Another solution is the EM algorithm, which was extended to work on the structure learning problem [3]. The idea is to use the E-step to compute the expected counts (which is a sufficient statistics for the problem) given the current structure of the network, and then apply such expected counts to learn a new structure. This is done iteratively until no improvement in the structure or parameters is possible. A possible version of the algorithm is as follows (there are other slightly different versions, e.g. step 2a may be replaced by a generalized EM):

1. Choose initial $\mathcal{G}^0$ and parametrization θ^0.
2. For $z = 0, 1, \ldots$ until there is no improvement:
   (a) Run the parameter EM method to find a new parametrization θ using the structure $\mathcal{G}^z$ and the parameters θ^z.
   (b) Run the structure learning method to find $\mathcal{G}^{z+1}$ and θ^{z+1} that maximizes the score function. Use the current expected counts of the variable from $\mathcal{G}^z$ and θ.

As long as the structure learning finds an improving solution, the method increases the score at each iteration, and so it converges. Until recently exact methods to find the best structure were simply prohibitive unless for toy examples, hence structure EM has been developed using a *Generalized EM* idea and step 2b was not global but an improving solution was enough. Still, if we use an approximate method such as hill climbing, we may decide to stop the search prematurely while in fact there was an improving solution that was not found by the approximate method. The use of a global method solves this situation, and we can prove that each iteration obtains a greater score until a stationary point is found. If an approximate method is used, one can only achieve the same result conditional on finding an improving during step 2b. The source of approximation in this enhanced structure EM is due solely to the use of the EM, which is intrinsic of the method. Therefore, this SEM procedure fully resembles the EM for learning parameters of networks of fixed structure. While the parameter EM has a closed form solution for the M-step, the structural learning idea presented here is not even polynomial. However, the B&B method is effective to find improving structures because of its *anytime* property, as we can stop it after finding one. As we will see in Section 4, the time is empirically dominated by the E-step and the differences between running only an approximate method are not large. Other exact methods [6, 7] are not anytime and each M-step would be very expensive.

4 Experiments

We use data sets of the UCI repository [1]. Continuous variables are discretized over the mean into binary variables. Furthermore, we introduce 20% of missing data (randomly chosen) in each data set: *adult* (15 vars and 30162 instances), *car* (7 vars and 1728 instances) *letter* (17 vars and 20000 instances), *mushroom* (23 vars and 1868 instances), *nursery* (9 vars and 12960 instances), Wisconsin Diagnostic Breast Cancer or *wdbc* (31 vars and 569 instances), *zoo* (17 vars and 101 instances).

Table 1. Comparison between methods on UCI data sets using BNs. 20% of the data are missing at random.

	BIC score		Time(min)	
Network	**B&B**	**HC**	**B&B**	**HC**
adult	-240415	-241017	188	185
car	-10458	-10888	3	3
letter	-187430	-188483	250	191
mushr	-10505.0	-11640.7	31	19
nursery	-95367.6	-95367.6	28	27
wdbc	-2953.1	-2964.2	43	28
zoo	-657.2	-685.5	1	1

Table 2. Comparison between methods on synthetic data using DBNs. 20% of the data are missing at random.

Method	BIC	Time	Mem
B&B	-2268.2	205.7sec	559.5KB
HC	-2311.1	197.8sec	80.4KB

Table 1 shows the results of SEM using the exact B&B method and the hill-climbing (HC) idea. Results of SEM with the B&B are better than those with HC in all but the *nursery* data set (equal in this case), which is case with the smallest search space by far. This is an expected behavior because large search spaces make the problem harder and the HC more susceptible to get trapped in a local maximum. Although the differences in the BIC score are not large, they happen constantly, which shows that it is usual to get trapped in local maxima. Eventually the differences might be large in some domains, as the method provides no guarantees. The extra cost of running the SEM with the B&B is affordable, at least for these data sets. It is worth noting that other exact methods, e.g. [7], have expected time of a single M-step for the data set *wdbc* of around four days.

For DBNs, we have run some experiments with synthetic data sets with 8 variables, 10 sequences and 50 slices per sequence, as shown in Table 2. The difference in time to run the SEM with B&B or hill-climbing is negligible, still the BIC score obtained with B&B is recurrently better (around 2% of improvement). The obtained results are in accordance with the results obtained for BNs, as expected. Overall, the gain is not large, but it comes without a significant computational cost.

5 Conclusions

This paper describes an idea to learn the structure of BNs and DBNs from incomplete data. We present a principled algorithm to translate the DBN learning problem in an equivalent BN learning problem. The exact B&B procedure guarantees global optimality of the BIC score and is anytime, which is specially important when we integrate it with an EM method to treat incomplete data sets. B&B ensures that the maximization step of EM is never trapped by a local optimum, and the anytime property allows the use of a generalized EM to reduce considerably the computational cost. Hence, the only source of approximation in this enhanced SEM method is intrinsic to the EM idea itself. We show in the experiments that our formulation has improved the SEM solutions when compared to a method that employs hill-climbing. As far as we know, this is the first attempt that brings the SEM method closer to the parameter EM method (in the sense of using an exact maximization step).

References

[1] A. Asuncion and D. Newman. UCI machine learning repository, 2007. http://www.ics.uci.edu/∼mlearn/MLRepository.html

[2] C. P. de Campos, Z. Zeng, and Q. Ji. Structure learning of Bayesian networks using constraints. In *International Conf. on Machine Learning*, pages 113–120, 2009.

[3] N. Friedman. The Bayesian Structural EM Algorithm. In *Conf. on Uncertainty in Artificial Intelligence*, pages 129–138, 1998.

[4] N. Friedman, K. Murphy, and S. Russell. Learning the structure of dynamic probabilistic networks. In *Conf. on Uncertainty in Artificial Intelligence*, pages 139–147, 1998.

[5] D. Heckerman, D. Geiger, and D. M. Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, 1995.

[6] P. Parviainen and M. Koivisto. Exact structure discovery in bayesian networks with less space. In *Conf. on Uncertainty in Artificial Intelligence*, 2009.

[7] T. Silander and P. Myllymaki. A simple approach for finding the globally optimal bayesian network structure. In *Conf. on Uncertainty in Artificial Intelligence*, pages 445–452, 2006.