Structural Learning in Object Oriented Domains

Olav Bangsø
Department of Computer Science
Aalborg University
Fredrik Bajers Vej 7E
9220 Aalborg, Denmark
bangsy@cs.auc.dk

Helge Langseth
Department of Computer Science
Aalborg University
Fredrik Bajers Vej 7E
9220 Aalborg, Denmark
hl@cs.auc.dk

Thomas D. Nielsen
Department of Computer Science
Aalborg University
Fredrik Bajers Vej 7E
9220 Aalborg, Denmark
tdn@cs.auc.dk

Abstract

When constructing a Bayesian network, it can be advantageous to employ structural learning algorithms to combine knowledge captured in databases with prior information provided by domain experts. Unfortunately, conventional algorithms do not exploit the occurrence of repetitive structures, which are often found in object oriented domains such as fault prediction in computer networks and large pedigrees.

In this paper we propose a method for structural learning in object oriented domains. It is demonstrated that this method is more efficient than conventional algorithms, and it is argued that the method supports a natural approach for expressing the prior information of domain experts.

Introduction

Bayesian Networks (BNs) (Pearl, 1988; Jensen, 1996) have established themselves as a powerful tool in many areas of artificial intelligence. However, one of the remaining obstacles is to create and maintain very large domain models. To remedy this problem, object oriented versions of the BN framework have been proposed in the literature, see e.g. (Koller and Pfeffer, 1997; Bangsø and Wuillemin, 2000). Although these frameworks relieve some of the problems when modeling large domains, it may still be difficult to elicit the parameters and the structure of the model. (Langseth and Bangsø, 2001) describes a method to efficiently learn the parameters in an object oriented domain, but the problem of specifying the structure still remains. Structural learning methods provide a way to combine an expert's knowledge with information from a database. Unfortunately, though, conventional structural learning algorithms, see e.g. (Heckerman et al., 1994), do not exploit that the domain may be object oriented.

This paper focuses on structural learning in object oriented domains, and a method for learning the structure of Object Oriented Bayesian Networks (OOBNs) is proposed. The OOBN framework provides an intuitive way of specifying prior knowledge for the structural learning algorithm. Furthermore, it is demonstrated that this learning method is more efficient than conventional learning methods.

Conventional structural learning

In what follows we outline the basis for performing conventional structural learning.

Let $\mathcal{D}$ denote a *database* of cases $\{\mathbf{C}_1, \ldots, \mathbf{C}_N\}$, where each case is a configuration $\mathbf{x}$ over a set of discrete variables $\mathbf{X} = (X_1, \ldots, X_n)$.

The BD metric

A Bayesian approach for measuring the fitness of a Bayesian network structure B_S, is its posterior probability given the database:

$$P(B_S|\mathcal{D}, \xi) = c \cdot P(B_S|\xi)P(\mathcal{D}|B_S, \xi),$$

where $c = 1/(\sum_B P(B|\xi)P(\mathcal{D}|B, \xi))$ and ξ is the prior knowledge. The normalization constant c does not depend on B_S, thus $P(\mathcal{D}, B_S|\xi) = P(B_S|\xi)P(\mathcal{D}|B_S, \xi)$ is usually used as the network score. We shall use the following notation to describe the qualitative properties of B_S: r_i is the number of states of X_i, q_i is the number of possible configurations over the parents of X_i and $\pi_i = j$ denotes the j'th configuration over the parents of X_i. For the quantitative properties, we use $\theta_{ijk} = P(X_i = k|\pi_i = j)$, $\Theta_{ij} = \cup_{k=1}^{r_i}\theta_{ijk}$, $\Theta_i = \cup_{j=1}^{q_i}\Theta_{ij}$ and $\Theta_{B_S} = \cup_{i=1}^{n}\Theta_i$.

The probability $P(\mathcal{D}, B_S|\xi)$ can be computed in closed form based on the following five assumptions: 1) the database $\mathcal{D}$ is a multinomial sample from some Bayesian network with parameters Θ_{B_S}, 2) the parameters Θ_{ij} are independent, 3) the database is complete, 4) the densities of the parameters Θ_{ij} depend only on the structure of the Bayesian network that is local to variable X_i (*parameter modularity*), and finally 5) the prior distribution of the parameters in every complete Bayesian network B_{S_c} has a Dirichlet distribution, i.e., there exist numbers (virtual counts) $N'_{ijk} > 0$ s.t.:

$$P(\Theta_{ij}|B_{S_c}, \xi) \propto \prod_k \theta_{ijk}^{N'_{ijk}-1}.$$

As mentioned above, $P(D, B_S|\xi)$ can now be computed in closed form given these five assumptions

(Cooper and Herskovits, 1992):

$$P(\mathcal{D}, B_S|\xi) = P(B_S|\xi) \quad \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(N'_{ij})}{\Gamma(N'_{ij}+N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(N'_{ijk}+N_{ijk})}{\Gamma(N'_{ijk})}, \qquad (1)$$

where Γ is the *Gamma* function satisfying $\Gamma(x+1) = x\Gamma(x)$ and N_{ijk} is the number of cases in $\mathcal{D}$ where $x_i = k$ and $\pi_i = j$; $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$ and $N'_{ij} = \sum_{k=1}^{r_i} N'_{ijk}$. This metric is known as the BD metric (Bayesian metric with Dirichlet priors). Unfortunately it requires the specification of the virtual counts N'_{ijk} for every complete Bayesian network structure.

The BDe metric

Another drawback of the BD metric is that networks which are *likelihood equivalent*, i.e. they encode the same assertions about conditional independence, need not be given the same score. Note that data cannot be used to discriminate between such networks. To overcome this problem, (Heckerman et al., 1994) describes the BDe metric (Bayesian metric with Dirichlet priors and equivalence) which gives the same score to likelihood equivalent networks. Furthermore, it provides a simple way of identifying the virtual counts in Equation 1.

The metric is based on the concept of sets of network structures. Each set is represented by a member B_S, and all members in a set satisfy the same independence assertions as B_S. Moreover, structures that belong to the same set are given the same score.

The virtual counts in Equation 1 can now be assessed by exploiting the notion of an *equivalent sample size* N':

$$N'_{X_1,\ldots,X_n} = P(X_1,\ldots,X_n|B_{S_c},\xi)\cdot N',$$

where $N' = \sum_{X_1,\ldots,X_n} N'_{X_1,\ldots,X_n}$. Thus, all the virtual counts can be found by constructing a BN for $\mathbf{X}$ and indicating the single value N'.

Finally, to calculate the score (Equation 1) we also need a prior probability $P(B_S|\xi)$ for the network structures. (Heckerman et al., 1994) uses the simple construction:

$$P(B_S|\xi) \propto \kappa^{\delta},$$

where $\delta = \sum_{i=1}^{n} \delta_i$, and δ_i denotes the number of parents for X_i that differs in the prior BN and B_S; each such parent is penalized by a constant $0 < \kappa \leq 1$. Note that this probability both captures the information from the prior BN and, if the prior network is sparse, penalizes the complexity of B_S.

The structural EM algorithm

In most real world problems we rarely have access to a complete database, hence Assumption 3 of the BD metric is likely to be violated. To accommodate those situations, (Friedman, 1998) describes the Structural EM (SEM) algorithm which basically "fills in" the missing values before searching the joint space of structures and parameters.

The SEM algorithm tries to maximize $P(\mathcal{D}, B_S|\xi)$, but instead of maximizing this score directly it maximizes the expected score. Let $\mathbf{o}$ be the set of observations from the database $\mathcal{D}$, and let $\mathcal{H}$ be the set of unobserved entries in $\mathcal{D}$. The general algorithm can then be outlined as:

Loop for $n = 0, 1, \ldots$ until convergence
a) Compute the posterior $P(\Theta_{B_S^n}|B_S^n, \mathbf{o})$.
b) **E-step:** For each B_S, compute:
$Q(B_S : B_S^n) = E[\log P(\mathcal{H}, \mathbf{o}, B_S)|B_S^n, \mathbf{o}]$.
c) **M-step:** Choose $B_S^{n+1} \leftarrow B_S$ that maximizes $Q(B_S : B_S^n)$.
d) **If** $Q(B_S^n : B_S^n) = Q(B_S^{n+1} : B_S^n)$ **then Return** B_S^n.

By exploiting linearity of expectation, (Friedman, 1998) derives an approximation for the summation in the E-step. Moreover, it is shown that by maximizing the expected score at each iteration we always obtain a better network in terms of its marginal score, $P(\mathbf{o}, B_S)$; this result also implies that the algorithm converges.

Object Oriented Bayesian Networks

In what follows the OOBN framework of (Bangsø and Wuillemin, 2000) will be outlined. The description is based on an example, that is also used to illustrate how structural learning is performed.

In the example a farm with two milk cows and two meat cows is modeled by an OOBN. These two types of cows are described by the classes Milk Cow and Meat cow. Following the object oriented idea, a superclass (Generic cow) for the cows is introduced to encapsulate their shared properties. The class Generic cow describes the general relationship between the food a cow eats, who the mother of the cow is and how much meat and milk it produces, see Figure 1. `Mother` and `Food` are *input nodes*; a reference to a node outside the class. `Milk` and `Meat` are *output nodes*; variables from a class that are usable outside the class. The sets of input nodes and output nodes are disjoint and form the interface of the class. A class may be instantiated several times with different nodes having influence on the different instantiations through the interface (e.g. the cows might have different mothers), so only the number of states of the input nodes is known. Moreover all instances of a class are assumed to be identical w.r.t. both parameters and structure. This assumption will be referred to as the *OO assumption*.

In the example, the classes Milk cow and Meat cow are both subclasses of the class Generic cow, so they must have the same or larger sets of input and output nodes as the class Generic cow. This ensures that a subclass can be used anywhere instead of one of its superclasses. Figure 2 illustrates the class Milk cow (an illustration of the class Meat cow is left out due to space limitations). All nodes in a subclass inherits the potentials of the corresponding nodes in its superclass unless they are overwritten. If a new parent is added to a node in a subclass, the inherited potential

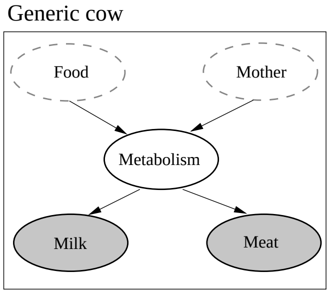

Figure 1: *The* Generic cow *class. The arrows are links as in normal BNs. The dotted ellipses are input nodes, and the shaded ellipses are output nodes.*

will be overwritten e.g. the potential for `Metabolism` is assigned a new parent in the class Milk cow (`State of mind`) so the potential is overwritten. It is, however, also possible to directly overwrite an inherited potential, e.g. the potential for `Metabolism` is assigned a new parent in the class Milk cow (`State of mind` so the potential is overwritten.

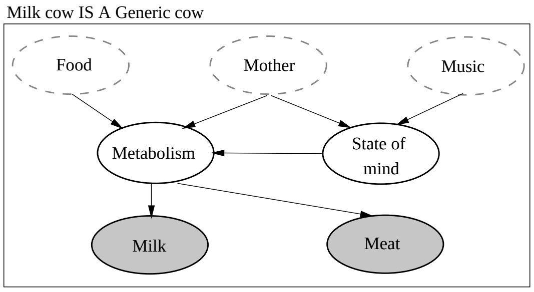

Figure 2: *The specification of the class* Milk cow*. Note that the input set is larger than the input set of the* Generic cow *(Figure 1).*

The live-stock of the farm is represented by the class Stock, in Figure 3, where the boxes indicate instances of a class e.g. Cow1 is an instance of the class Meat cow. Note that only input nodes and output nodes are visible, as they are the only nodes of an instance that are available to the encapsulating class (Stock). The double arrows (termed *reference links*) indicate that an input node is referencing a node outside the instantiation, e.g. the input node `Mother` of Cow1 references the node `Daisy`. The direction of a reference link implies that the parent is used in the instantiation. Notice that not all of the `Mother` nodes in the Stock specification are referencing a node. To ensure that these nodes are associated with a potential, the notion of a *default potential* is introduced. A default potential is a probability distribution over the states of an input node, and is used when the input node is not referencing any node. Inference can be performed by compiling the OOBN into a *multiply-sectioned Bayesian network* (Xiang et al., 1993), or by constructing the *underlying BN*, see (Bangsø and Wuillemin, 2000) for details.

Structural Learning in OOBNs

When performing structural learning, it is expedient to take advantage of prior knowledge about the structure of the domain. For example, it is often the case that a domain expert is able to partition the variables of the domain into subsets that have high internal coupling, and low external coupling. In addition the domain expert may be able to identify certain subsets representing different entities with the same properties. This closely resembles a partitioning of the domain variables into instances, where instances with the same properties belong to the same class.

A natural requirement for structural learning in an object oriented domain is that given prior knowledge (represented by a partial OOBN model) and data about the domain, the resulting model should still be an OOBN. This is, however, not ensured by conventional learning methods.

Another aspect of structural learning is the complexity of the search space. The complexity varies with the prior knowledge, and in this paper we consider two distinct levels of prior knowledge. Notice that this prior knowledge concerns the object orientation of the domain, and it is therefore not easily exploited in a conventional setting. This implies that the proposed method is expected to produce better results than conventional learning, since a reduction of the search space reduces the number of local maxima encountered. The proposed method learns from complete (or completed) data, so each instance is d-separated from the rest of the network, given its interface (Bangsø and Wuillemin, 2000). Hence, the structure inside each class can be learned locally in the class.

To evaluate a candidate structure during model search, the parameters in the model must be estimated. We utilize the object-oriented learning method by (Langseth and Bangsø, 2001), where cases from all instances of a class are regarded as (virtual) cases of the class. That is, learning is performed in the class specifications, and not in each instantiation. However, this requires that it must be known what parents a node has, in each instance where it appears. The search for a good set of parents is not necessarily local to a class when the interface is not given. Consider Cow3 and Cow4 that are both instances of the class Milk cow. If Cow3.`Metabolism` gets a new parent during model search, Cow4.`Metabolism` must be assigned a new parent as well (due to the OO assumption). If Cow3.`Metabolism` is assigned an input node as parent and that input node references `Weather`, the search algorithm should find a node Z that has the same influence on Cow4.`Metabolism` (by way of the input node) as `Weather` has on Cow3.`Metabolism` (given the current model). The search for Z must therefore cover all output nodes in all instances in Stock, as well as all nodes in the encapsulating class, Stock. Note that Z may be non-existent in which case the default potential for the input node is used. The complexity of finding the best candidate for all instances is exponential in the

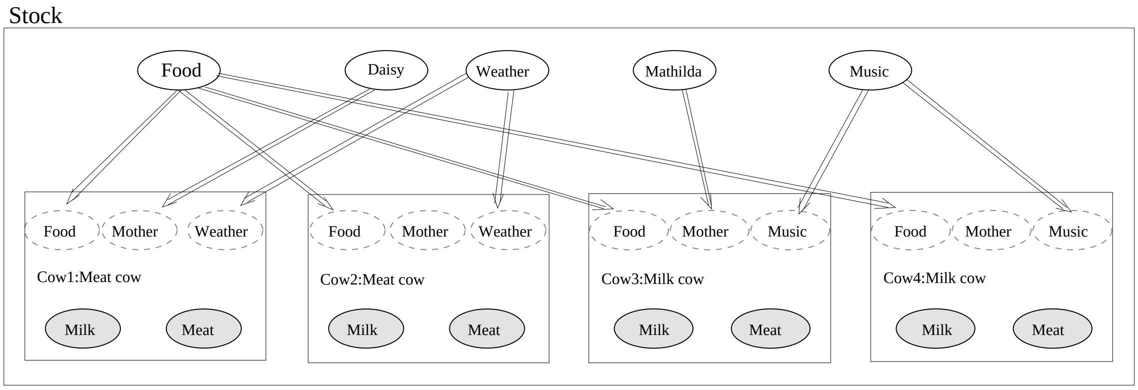

Figure 3: *The* STOCK *with two* MILK COW*s and two* MEAT COW*s. Note that some input nodes are not referencing any nodes.*

number of instances, and we risk using a non-negligible amount of time to evaluate models with low score.

The following is a description of the two levels of information, which is considered.

Plain object oriented learning. The modeler indicates the instance each node belongs to and whether or not it is a possible output node. Each instance is associated with a class as well.

With this information, all possible nodes should be investigated to see if they should be referenced by any of the input nodes of each instance.

Restricted input sets. Again the modeler indicates the instance each node belongs to and whether or not it is allowed to be an output node. The class of each instance is indicated, as well as the nodes each instance is allowed to reference (e.g. only `Daisy`, `Mathilda`, `Weather`, `Music` and `Food` can be referenced by the instances of MILK COW and MEAT COW). This yields a restriction on the search space compared to plain object oriented learning.

In practice the information gained from a domain expert will be a mixture of these information levels. This can easily be accommodated in our method, as plain object oriented learning can be seen as a case of restricted input sets, where all possible nodes are allowed to be referenced. This issue will not be discussed further in this paper, as the performance of such a mixture will clearly be somewhere between the two information levels.

Finally, the expert might also be able to specify exactly what nodes are referenced by each instance, and which nodes in the instances these are parents of. In that case, investigation regarding references is not necessary; all that remains is to identify the structure inside the classes.

Empirical results

In this section we will investigate the merits of the proposed learning algorithm by employing it to discover the OOBN representation of the STOCK domain. That is, we generate data describing the domain and then use it to learn an OOBN model, as outlined in the previous section.

The goal of the empirical study is to evaluate whether or not the proposed learning method generates a good estimate of the unknown statistical distribution (as opposed to causal inference where the directions of the arcs are of interest). Let $f(\mathbf{x}|\Theta)$ be the unknown *gold standard distribution* encoded by the STOCK class; $\mathbf{x}$ is a configuration of the domain and Θ are the parameters. $\hat{f}_N(\mathbf{x}|\hat{\Phi}_N)$ will be used to denote the approximation of $f(\mathbf{x}|\Theta)$ based on N cases from the database.

Since an estimated model may have other links than the gold standard model, the learned CPTs of $\hat{\Phi}_N$ may have other domains than the CPTs of Θ. Hence a global measure for the difference between the estimated model and the gold standard model is required. In the tests performed we have measured this difference by using the empirical Kullback-Leibler (KL) divergence between the estimated model and the gold standard model. The calculations were performed in the underlying BNs of the two models. The KL divergence is defined as

$$I(\hat{f}, f) = \sum_{\mathbf{x}} \hat{f}_N(\mathbf{x}|\hat{\Phi}_N) \log \left[\frac{\hat{f}_N(\mathbf{x}|\hat{\Phi}_N)}{f(\mathbf{x}|\Theta)} \right].$$

This can be calculated without expanding the sum as long as $\log[\hat{f}_N(\mathbf{x}|\hat{\Phi}_N)/f(\mathbf{x}|\Theta)]$ factorizes over the cliques of the junction tree (JT) representation of $\hat{f}_N$, see (Cowell et al., 1999, Chapter 6). Care must be taken to ensure that the cliques of the estimated model are large enough to capture all the dependences in the gold standard model. This can be achieved by compiling the estimated model such that if X and Y are in the same clique in the JT-representation of the gold standard model, then they are also in the same clique in the JT-representation for the estimated model.

Note that if the causal interpretation of the generated model had been of interest, the KL divergence would not have been a well suited divergence measure; a direct measure of structural difference is more appropriate in that case.

As described in the previous section we consider two different levels of information available to the learning algorithm; plain object oriented learning and learning with restricted input sets. For comparison, we also employed a conventional learning algorithm.

The learning method was tested by randomly generating a database of size N from the gold standard model, where 25% of the data was missing. The database was used as input to the structural learning algorithm. This was repeated a total of 100 times, with N varying from 100 to 10.000. Each level of information was used to generate models for each data set, and the KL divergence was then calculated.

In our tests we used the SEM-algorithm with a maximum of ten iterations (convergence was typically reached in 4–5 iterations). In each iteration a simulated annealing search was performed over all models consistent with the current level of input. The parameters in the simulated annealing were $T_0 = 25$, $\alpha = 75$, $\beta = 75$, $\gamma = 0.75$ and $\delta = 100$, see (Heckerman et al., 1994) for notation. The empty graph was used as our prior network, and it was given an equivalent sample size of $N' = 64$.

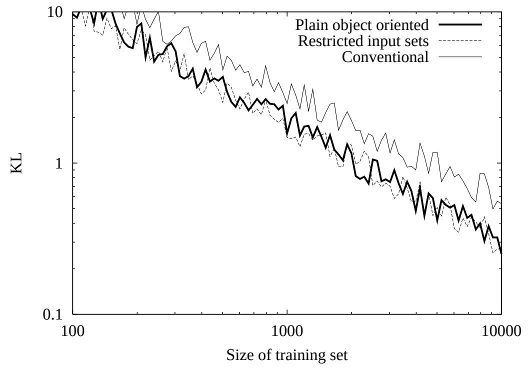

Figure 4: *The KL divergence of the generated models vs. the gold standard model. The different lines correspond to the different levels of information available to the learning algorithm.*

The empirical results for the STOCK domain is given in Figure 4. As can be expected, the quality of the result increases with the level of information available to the learning algorithm. Note that by just enforcing the OO assumption the quality of the learned network increases by a factor of approximately 1.6 (by going from conventional to plain object oriented learning). Surprisingly the gain from restricting the input sets is only marginal in the example (an improvement by a factor of 1.03). The restriction imposed in our test runs was that a node in the output set of one instance could not be in the input set of another. It seems that this knowledge was easily discovered by the learning algorithm (even for small data sets).

Getting to know the correct reference nodes of each instance improves the results from the restricted input set learning by a factor of approximately 1.7. These values are fairly constant as the size of the database increases, thus the quality of the network approximation increases uniformly; the algorithm proposed in this paper is just as good at learning reference nodes as it is at learning the other parts of the domain, even though the calculations are more complex.

Conclusion

In this paper we have proposed a method for doing structural learning in object oriented domains. The learning algorithm is based on the OOBN framework by (Bangsø and Wuillemin, 2000), and has been implemented using the Structural EM algorithm by (Friedman, 1998).

The proposed learning algorithm exploits an intuitive way of expressing prior information in object oriented domains, and it was shown to be more efficient than conventional learning algorithms in this setting.

References

Bangsø, O. and Wuillemin, P.-H. (2000). Object Oriented Bayesian networks. A Framework for Topdown Specification of Large Bayesian Networks with Repetitive Structures. Technical report CIT-87.2-00-obphw1, Department of Computer Science, Aalborg University.

Cooper, G. F. and Herskovits, E. (1992). A Bayesian Method for the Induction of Probabilistic Networks from Data. *Machine Learning*, 9:309–347.

Cowell, R. G., Dawid, A. P., Lauritzen, S. L., and Spiegelhalter, D. J. (1999). *Probabilistic Networks and Expert Systems.* Springer.

Friedman, N. (1998). The Bayesian Structural EM Algorithm. In *Proc. UAI-98*. Morgan Kaufmann.

Heckerman, D., Geiger, D., and Chickering, D. M. (1994). Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. In *Proc. UAI-94*, pages 293–301. Morgan Kaufmann.

Jensen, F. V. (1996). *An introduction to Bayesian networks.* UCL Press.

Koller, D. and Pfeffer, A. (1997). Object-oriented Bayesian networks. In *Proc. UAI-97*, pages 302–313. Morgan Kaufmann.

Langseth, H. and Bangsø, O. (2001). Parameter Learning in Object Oriented Bayesian Networks. *Annals of Mathematics and Artificial Intelligence.* To appear.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann.

Xiang, Y., Poole, D., and Beddoes, M. P. (1993). Multiply sectioned Bayesian networks and junction forests for large knowledge-based systems. *Computational Intelligence*, 9(2):171–220.