Using Feature Hierarchies in Bayesian Network Learning (Extended Abstract)

Marie desJardins[1], Lise Getoor[2], and Daphne Koller[2]

[1] AI Center
SRI International
marie@ai.sri.com
[2] Computer Science Department
Stanford University
getoor@cs.stanford.edu, koller@cs.stanford.edu

1 Introduction

In recent years, researchers in statistics and the UAI community have developed an impressive body of theory and algorithmic machinery for learning Bayesian networks from data. Learned Bayesian networks can be used for pattern discovery, prediction, diagnosis, and density estimation tasks. Early pioneering work in this area includes [5, 9, 10, 13]. The algorithm that has emerged as the current most popular approach is a simple greedy hill-climbing algorithm that searches the space of candidate structures, guided by a network scoring function (either Bayesian or Minimum Description Length (MDL)-based). The search begins with an initial candidate network (typically the empty network, which has no edges), and then considers making small local changes such as adding, deleting, or reversing an edge in the network.

Within the context of Bayesian network learning, researchers have examined how to use background knowledge about the network structure [8, 14] to guide search. The most commonly used form of background knowledge is information about variable ordering, which is often inferred from temporal information about the domain, and translates into constraints on edge directions. Other types of background knowledge include constraints on edge existence and constraints on whether a node must be a root or may have parents.

Researchers have also examined the task of discretizing variable values in Bayesian network learning [6, 11, 15]. The objective is to find an appropriate discretization of a continuous variable, or an appropriate partitioning for an ordinal variable, that will lead to a higher-scoring network (which one hopes, in turn, will translate into improved generalization performance).

Another line of research has investigated how constraints on local probability models within the Bayesian network can be represented using either decision trees [2, 7] or decision graphs [4]. These approaches provide a compact representation for the conditional probability table (CPT) that is associated with each node in the network. One way to view these methods is that they partition a CPT into contexts in which the conditional probabilities are equal.

In this abstract, we describe an approach that draws from the above three lines of research. We examine the use of background knowledge in the form of feature hierarchies during Bayesian network learning. Feature hierarchies enable us to aggregate categorical variables in meaningful ways. This allows us to choose an appropriate "discretization" for a categorical variable. In addition, by choosing the appropriate level of abstraction for the parent of a node, we also support compact representations for the local probability models, thus encoding constraints on the contexts in which conditional probabilities are equal.

Our hypothesis is that using feature hierarchies will enable us to learn networks that have better generalization performance, because we can learn a network where each parent node is at the appropriate level of abstraction, and can in fact be at different levels of abstraction in different contexts. The resulting networks are more compact, require fewer parameters, and capture the structure of the data more effectively.

We begin with a brief overview of Bayesian networks. We then describe Abstraction-Based Search (ABS), a Bayesian network learning algorithm we have developed that makes use of feature hierarchies, and present preliminary experimental results in several domains.

2 Bayesian Networks

Bayesian networks [12] are a compact representation of a joint distribution over a set of random variables, $X_1, \ldots, X_n$. Bayesian networks utilize a structure that exploits conditional independences among variables, thereby taking advantage of the "locality" of probabilistic influences. The first component is a directed acyclic graph whose nodes correspond to the random variables $X_1, \ldots, X_n$, and whose links denote direct dependency of a variable X_i on its parents $\mathrm{Pa}(X_i)$. Given the graph component, the second component describes the quantitative relationship between the node and its parents as a *conditional probability table (CPT)*, which specifies the distribution over the values of X_i for each possible assignment of values to the variables in $\mathrm{Pa}(X_i)$. The conditional independence assumptions associated with the dependency graph, together with the CPTs associated with the nodes, uniquely determine a joint probability distribution over the random variables.

The problem of learning a Bayesian network from a data set can be stated as follows. Given a *training set* D of independent instances, find a network that *best matches* D. The common approach is to introduce a statistically motivated scoring function that evaluates how well the network matches the training data, and to search for the optimal network according to the score. A widely used scoring function is the *Bayesian score* [5, 9]:

$$P(B_S|D) \propto P(D|B_S)P(B_S)$$

Given complete data, and making certain assumptions about the process that generates the data and the form of the priors for the CPT entries, it is possible

to derive a closed-form solution for the score of a candidate structure B_S:

$$P(B_S)\prod_i \prod_{\mathrm{pa}(X_i)} \frac{\Gamma(\alpha_{\mathbf{pa}(X_i)})}{\Gamma(N_{\mathbf{pa}(X_i)}+\alpha_{\mathbf{pa}(X_i)})}\prod_{x_i}\frac{\Gamma(N_{x_i,\mathbf{pa}(X_i)}+\alpha_{x_i,\mathbf{pa}(X_i)})}{\Gamma(\alpha_{x_i,\mathbf{pa}(X_i)})}$$

where $\mathrm{pa}(X_i)$ are possible instantiations for the parents of X_i, the α values characterize our prior information about each parameter in the network, and N are the counts in the data D for a particular instantiation of the variables. There are a number of possible choices for the network prior for $P(B_S)$, but a typical requirement is that it can be factored into products of functions that depend only on a node and its parents. Common choices include a uniform prior and a prior that favors networks with fewer parameters. For further details about scoring functions, [8] and [9] are excellent resources.

The problem of finding a network that optimizes this score is NP-hard [3], so we resort to heuristic search. Surprisingly, a simple greedy hill-climbing search is often quite effective. The search algorithm has local operators Add(X, Y), which adds X as a parent of Y, Delete(X, Y), which deletes X from the parents of Y and Reverse(X, Y), which reverses an edge from X to Y, making Y a parent of X. When there are no missing values in the data, the scoring function decomposes locally, so that when one of these operators is applied, only the score at the node whose parents have changed needs to be recomputed. Exploiting this property allows the scores of alternate structures to be computed efficiently.

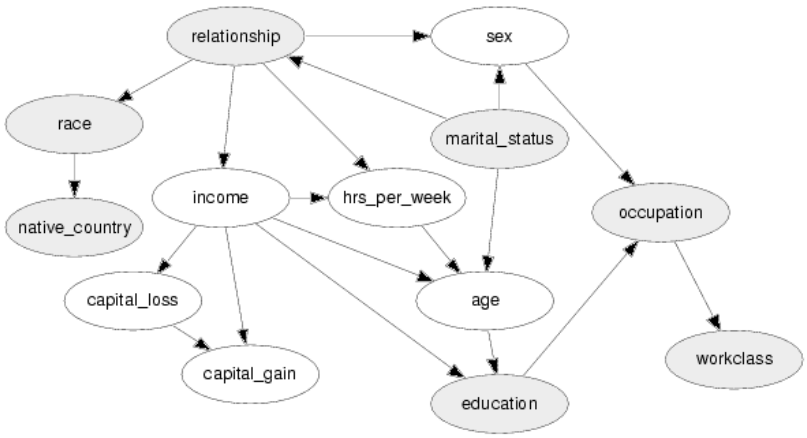


Fig. 1. A Bayesian network for the census domain. The shaded nodes have feature value hierarchies.

Figure 1 shows a Bayesian network for a census domain. Part of the CPT for *Occupation* is shown in Table 1. Each row in the CPT corresponds to a potential

Table 1. CPT for *Occupation* in the Bayesian network of Figure 1

	Occupation						
Education, Sex	Craft/ repair	...	Exec/ mgr.	Prof. specialty	...	Protective services	Armed forces
Preschool, Female	0.06	...	0.06	0.06	...	0.06	0.06
Preschool, Male	0.082	...	0.034	0.034	...	0.034	0.034
1st_4th, Female	0.03	...	0.03	0.11	...	0.03	0.03
1st_4th, Male	0.25	...	0.017	0.017	...	0.017	0.017
...							
Masters, Female	0.01	...	0.19	0.65	...	0.0042	0.0042
Masters, Male	0.0025	...	0.32	0.43	...	0.02	0.0018
Doctorate, Female	0.054	...	0.085	0.55	...	0.022	0.022
Doctorate, Male	0.017	...	0.14	0.73	...	0.0073	0.0073

instantiation of the parents, in this case *Education* and *Sex*. A given entry in the CPT specifies the probability that *Occupation* takes on the value corresponding to that column, given the values for *Education* and *Sex* associated with that row. The full CPT has 32 rows (16 education levels and two values for *Sex*) and 14 columns (occupation categories), resulting in a total of 448 table entries. Since each of these parameters must be separately estimated by the learning algorithm, it is apparent that using abstraction to compress the size of the CPT may result in better parameter estimation, thus improving learning performance.

3 Learning Bayesian Networks Using Feature Hierarchies

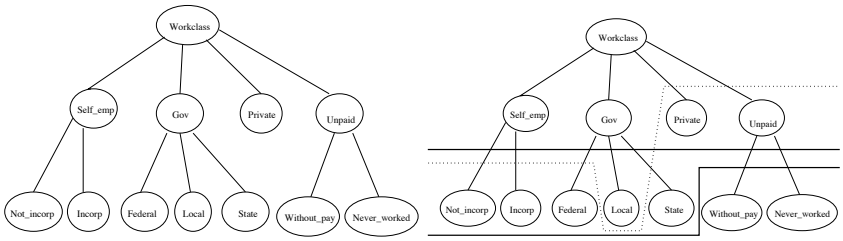
We describe how to extend existing methods for learning Bayesian networks to make use of background knowledge in the form of feature hierarchies. We begin by discussing feature hierarchies in more detail and then describe the learning algorithm.

3.1 Feature Hierarchies

A feature hierarchy defines an IS-A hierarchy for a categorical feature value. The leaves of the feature hierarchy describe base-level values—these are the values that occur in the training set.[1] The interior nodes describe abstractions of the base-level values. The intent is that the feature hierarchy is designed to define useful and meaningful abstractions in a particular domain.

Figure 2(a) shows a feature hierarchy for *Workclass* in the census domain, which describes an individual's employer type. At the root, all workclass types are grouped together. Below this are three abstract workclass values—Self-emp,

[1] This is not a strict requirement: with appropriate additional assumptions, we can allow training instances that are described at abstract levels, although we do not investigate this possibility here.



(a) Feature value hierarchy (b) Abstraction levels

Fig. 2. (a) *Workclass* feature hierarchy (b) Legal (solid) and illegal (dotted) abstraction levels

Government, and Unpaid—and one base-level value, Private. Each of the abstract values is further subdivided into the lowest-level values that appear in the raw data. As shown by this example, a feature hierarchy need not be balanced (i.e., path length from leaf nodes to the root can vary), and the branching factor (number of children) can vary from one node to another.

A cut through the tree defines an *abstraction level*, which is equivalent to a mutually exclusive and disjoint set of abstract feature values. Figure 2(b) shows three different abstraction levels for *Workclass*. Each abstraction level contains the set of nodes immediately above the cut line. The solid lines correspond to *legal* abstraction levels. The upper abstraction level includes the values Self-emp, Gov, Private, and Unpaid. The lower abstraction level includes the values Not-incorp, Incorp, Federal, Local, State, Private, and Unpaid. In this case, the lower abstraction level makes more distinctions than the upper abstraction level. The dotted line corresponds to an *illegal* abstraction level: it includes both Gov and Local, which are not mutually exclusive.

The feature hierarchy helps to bias our search over appropriate abstractions for a categorical variable. Without the hierarchy to guide us, we would need to consider arbitrary subsets of the base-level values for abstractions. Here, the feature hierarchy tells us which combinations of the values are meaningful (and, hopefully, useful in density estimation).

3.2 Learning Algorithm

There are two key tasks to be performed when learning a probabilistic model: scoring a candidate model and searching the space of possible models. It is straightforward to extend the scoring functions for nodes modeled at different levels of abstraction. For example, for the score described earlier, we simply marginalize over the appropriate base-level values to compute N, the counts in the data, and α, the prior, for the abstract values of a given variable.

ABS extends the standard search over network structures as follows. When an edge is added to the network, the parent is added at its most abstract level. For example, if *Workclass* is chosen as a parent, the initial abstraction level would be {Self-emp, Gov, Private, Unpaid} (the upper abstraction level in Figure 2(b)).

ABS extends the standard set of BN search operators—edge addition, edge deletion, and edge reversal—with two new operators that can refine an edge or abstract an edge. The search process is a greedy search algorithm that repeatedly applies these five operators to the current network, evaluates the resulting network using the Bayesian score, and replaces the current network with the new one if the latter outscores the former.

The new operators are Refine(X, Y, i) and Abstract(X, Y, i). If X is the parent of Y, and its current abstraction level is $\{v_1, \ldots, v_k\}$, Refine(X, Y, i) refines the ith value of the abstraction, v_i, by replacing v_i with the set of values of its children in the feature hierarchy. During the search process, ABS attempts to apply Refine to each value of each abstraction in the current network. Refine only succeeds if the value it is applied to is an abstract value (i.e., if the value has children in the feature hierarchy).

Similarly, if X is the parent of Y, and its current abstraction level is $\{v_1, \ldots, v_k\}$, Abstract(X, Y, i) abstracts v_i by replacing v_i and its siblings with the value of their parent in the feature hierarchy. Again, during search, ABS attempts to apply Abstract to each value of each abstraction level. Abstract only succeeds if the parent value is below the root node of the feature hierarchy and all of the value's siblings appear in the abstraction level. For example, in the lower abstraction level shown in Figure 2(b), neither condition is satisfied for the value Unpaid: its parent value is the root node of the hierarchy, and Unpaid's siblings Self-emp and Gov do not appear in the abstraction level.

Several examples of legal applications of Abstract and Refine are given in Table 2. The boldface values are those that are changed by the operation.

Table 2. Examples of Abstract and Refine operators

Initial abstraction level	Operation	Final abstraction level
{**Self-emp**, Gov, Private, Unpaid}	Refine ($Workclass, Y, 1$)	{**Not-incorp, Incorp**, Gov, Private, Unpaid}
{Not-incorp, Incorp, **Gov**, Private, Unpaid}	Refine ($Workclass, Y, 3$)	{Not-incorp, Incorp, **Federal, Local, State**, Private, Unpaid}
{**Not-incorp, Incorp**, Federal, Local, State, Private, Unpaid}	Abstract ($Workclass, Y, 1$)	{**Self-emp**, Federal, Local, State, Private, Unpaid}

4 Results

In this section, we describe initial results on three domains: a synthetic Bayesian network (Synthetic) and two real-world domains, U.S. census data (Census) and

tuberculosis patient data (TB). We present results for ABS and for FLAT, a learning algorithm that does not use the feature hierarchies.

4.1 Test Domains

The synthetic network has 20 random variables, each with a domain consisting of four discrete values. Five of the variables have feature hierarchies associated with them, each of which structures the four values for the node into a 3-level binary hierarchy. The CPTs for each node were filled in randomly, using the middle level of the hierarchy (i.e., the two aggregated values) for each of the hierarchical nodes. We then generated a training set and a test set from the network. Because we have the original network as a "gold standard," we can measure the distance from the learned network to the true network, as well as evaluating the score of the learned network and its performance on the test set.

For the second set of experiments, we used the census domain described in Section 2. There are feature hierarchies for seven of the nominal variables: work category, education, marital status, occupation, relationship, race, and native country. (These nodes are shaded in Figure 1.) Figure 3 shows the feature hierarchy for *Education*, which includes 16 values in the raw data, ranging from Preschool through Doctorate. There are three abstract values: No-HS (grouping all levels below high school graduate), Post-HS (high school degree or more, but no college degree), and Post-College (graduate degree).

For our third set of experiments, we used a database of epidemiological data for 1300 San Franciscan tuberculosis (TB) patients [1]. There are 12 variables in this dataset, including patient's age, gender, ethnicity, place of birth, and medical history (HIV status, disease site, X-ray result, etc.). We constructed feature hierarchies for two of the variables: place of birth and ethnicity.

4.2 Experiments

Table 3 shows results for each of these three domains. The column labelled "Network Score" shows the scores of the networks learned in each domain by the ABS and FLAT algorithms. The results are for a set of 100 runs in each domain. In each case, we see that the mean score of the network learned by ABS is slightly better than that of the FLAT network. While the difference in score is not large, it is statistically significant at well over the 99% confidence interval range.

While we are interested in finding higher-scoring networks, we are more interested in improved performance on unseen data. This tests whether using feature abstractions results in improved generalization performance. The column labelled "Log-Likelihood of Test Set" shows the mean log-likelihood of the test set for each domain for both FLAT and ABS for 100 runs. For each of the domains, the likelihood of the test set according to the network learned by ABS is better than for the FLAT network. Again, these results are statistically significant with over 99% confidence.

Table 3. Log-likelihood of test sets and scores for ABS and FLAT on three domains: Census, TB and Synthetic. In all cases, ABS outperforms FLAT at confidence intervals over 99%.

			Log-Likelihood of Test Set					Network Score				
			ABS		FLAT			ABS		FLAT		
Domain	Train	Test	Mean	Std	Mean	Std	CI	Mean	Std	Mean	Std	CI
Census	10000	5000	-9.85	0.064	-9.86	0.065	99%	-9.59	0.047	-9.59	0.047	99%
Census	15000	5000	-9.33	0.057	-9.34	0.057	99%	-8.58	0.037	-8.59	0.036	99%
Census	20000	5000	-8.38	0.064	-8.41	0.063	99%	-7.61	0.029	-7.62	0.030	99%
Census	25000	5000	-5.72	0.05	-5.74	0.05	99%	-6.66	0.027	-6.67	0.027	99%
TB	1000	200	-5.09	0.22	-5.10	0.22	99%	-5.49	0.10	-5.54	0.10	99%
Synthetic	1000	500	-23.71	0.20	-23.85	0.20	99%	-23.51	0.19	-23.76	0.19	99%

As we mentioned earlier, for the synthetic domain, we can also compute the KL-distance from the learned networks to the gold-standard network. For a training set of size 10,000, the distance for FLAT is 0.38 while the distance for ABS is much better at 0.28.

4.3 Characteristics of Learned Networks

It is interesting to examine more carefully the differences between the learned networks. Figure 1 shows a Bayesian network learned in the census domain using FLAT while Figure 3 shows a Bayesian network learned using ABS. The shaded nodes in Figure 3 indicate the variables that have parents that have been abstracted.

Particularly for the variables with large domain sizes (such as *Native-country*, *Occupation*, and *Education*), ABS is successful in finding abstraction levels that reduce the size of the CPT. In this network, the abstraction level for *Education* in the CPT for the edge from *Education* to *Native-country* includes only four values rather than 16. The resulting abstraction level is shown in the lower right of Figure 3. Similarly, for the edge from *Occupation* to *Education*, the abstraction level for *Occupation* includes five values rather than the 14 base-level values.

The effect of this CPT compression is that edges are often added into the network that are not added by the FLAT algorithm. One way to view this is that we can make better use of our parameter resources, capturing only relevant data dependencies for each parent, by modeling them at the appropriate level of abstraction. Because we are not "wasting" parameters by modeling unnecessary distinctions, we can model more dependencies, more effectively, with the same amount of data. The final network in the FLAT case contains 19 links, whereas the final network learned by ABS contains 23 links. Although the ABS graph structure is significantly more complex, the CPTs are compressed and therefore contain only one-third more parameters (2032 parameters) than the FLAT network (1542 parameters). If the ABS network were represented without any

compression, it would contain 4107 parameters. Thus, in this case, abstraction yields a "parameter savings" of over 50%.

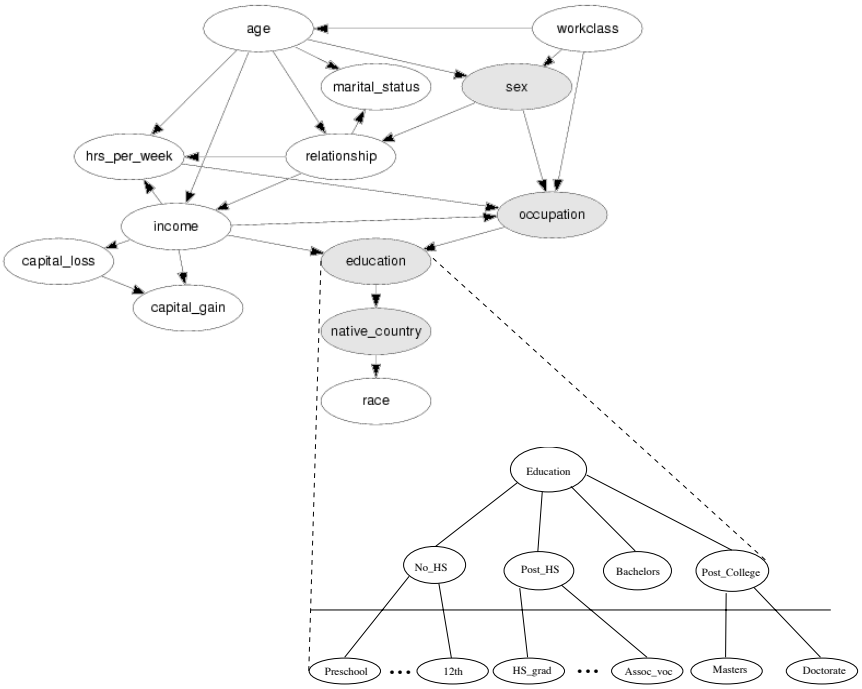


Fig. 3. A Bayesian network learned by ABS for the census domain. The shaded nodes have parents that have been abstracted. The abstraction level for *Education* is also shown.

5 Related Work

ABS can be thought of as a way to aggregate nominal variables. Many researchers have explored the problem of discretizing continuous variables, and aspects of this research are relevant to our approach. Friedman and Goldszmidt [6] present a Minimum Description Length (MDL)-based approach for discretizing variables during Bayesian network learning. Monti and Cooper [11] use a latent variable model to perform multivariate discretization. Wellman and Liu [15] describe a statistical approach for creating abstract variable values by combining neighboring values of ordinal variables in a Bayesian network—essentially discretization via aggregation for integer-valued variables.

Tree-Structured CPTs (TCPTs) and other representations for CPTs that take advantage of local regularities within the probabilistic model provide a

compact representation of the local probability models and, when used during learning, can result in higher-scoring networks [2, 4]. These approaches represent context-specific independence (CSI), that is, independencies that hold when certain variables take on certain values.

TCPTs represent the CPT as a decision tree. Each branch of the tree is associated with a particular value assignment for one of the parent variables of a node in a Bayesian network. Each leaf node therefore defines a *context* represented by the variable assignments along the path from the root to that leaf. The probability stored at the leaf node is the probability to be used for any assignment of values that is consistent with that context.

TCPTs and ABS represent complementary approaches to the problem of efficiently representing CPTs within a Bayesian network. TCPTs provide a representation for describing CSI relationships. ABS, on the other hand, defines a search space in which each point is an abstraction level that constrains the set of possible CSI relationships. ABS could be combined with CSI by using the feature-hierarchy-based search to identify appropriate tests within a TCPT.

6 Conclusions and Future Work

We have presented preliminary results which show that using feature hierarchies during learning can result in better scoring networks that also have better generalization performance. While this is a useful extension to traditional Bayesian network learning algorithms, it is not a particularly surprising result. Perhaps more surprising is that while ABS always outperforms FLAT, the performance improvement is not as large as we would expect. We are interested in further improving performance by exploiting feature abstraction in other ways during learning. In addition to the ABS approach for incorporating feature value hierarchies into Bayesian network structure learning, we have developed two methods for parameter estimation using these hierarchies. One method uses statistical hierarchical modeling methods to estimate the parameters in the CPT; the other uses weight-based mixture modeling. We plan to compare these approaches to the ABS approach to determine which method (or combination of methods) works the best in practice.

We are also interested in comparing these feature-hierarchy-based approaches with learning methods that create structured CPTs, such as TCPTs. ABS could be combined with TCPTs by using the feature-hierarchy-based search to identify appropriate tests within a TCPT.

Finally, if a feature value hierarchy is not available a priori, it would be possible to apply clustering techniques within the search algorithm to find appropriate sets of value groupings in the network. It would be interesting to develop such clustering methods and compare their performance to the alternatives (flat learning or ABS with a known feature value hierarchy).

Acknowledgements

Thanks to Peter Small and Jeanne Rhee of the Stanford Medical School for providing the tuberculosis patient data. Partial support for this work was provided by DARPA's High-Performance Knowledge Bases program.

References

[1] M.A. Behr, M.A. Wilson, W.P. Gill, H. Salamon, G.K. Schoolnik, S. Rane, and P.M. Small. Comparative genomics of BCG vaccines by whole genome DNA microarray. *Science*, 284:1520–23, 1999.

[2] Craig Boutilier, Nir Friedman, Moises Goldszmidt, and Daphne Koller. Context-specific independence in Bayesian networks. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence (UAI-96)*, pages 115–123, August 1996.

[3] D. M. Chickering. Learning Bayesian networks is NP-complete. In D. Fisher and H.-J. Lenz, editors, *Learning from Data: Artificial Intelligence and Statistics V*. Springer Verlag, 1996.

[4] D. M. Chickering, D. Heckerman, and C. Meek. A Bayesian approach to learning Bayesian networks with local structure. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pages 80–89, 1997.

[5] G. F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

[6] Nir Friedman and Moises Goldszmidt. Discretizing continuous attributes while learning Bayesian networks. In *Proceedings of the Thirteenth International Conference on Machine Learning*, 1996.

[7] Nir Friedman and Moises Goldszmidt. Learning Bayesian networks with local structure. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence (UAI-96)*, 1996.

[8] D. Heckerman. A tutorial on learning with Bayesian networks. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer, Dordrecht, Netherlands, 1998.

[9] D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995.

[10] W. Lam and F. Bacchus. Learning Bayesian belief networks: An approach based on the MDL principle. *Computational Intelligence*, 10:269–293, 1994.

[11] Stefano Monti and Gregory F. Cooper. A latent variable model for multivariate discretization. In *Proceedings of the Seventh International Workshop on AI & Statistics (Uncertainty 99)*, 1999.

[12] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Francisco, 1988.

[13] D. J. Spiegelhalter, A. P. Dawid, S. L. Lauritzen, and R. G. Cowell. Bayesian analysis in expert systems. *Statistical Science*, 8:219–283, 1993.

[14] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction and Search*. Number 81 in Lecture Notes in Statistics. Springer-Verlag, NY, 1993.

[15] Michael P. Wellman and Chao-Lin Liu. State-space abstraction for anytime evaluation of probabilistic networks. In *Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence (UAI-93)*, pages 567–574, 1994.