LEARNING BAYESIAN NETWORKS WITH LOCAL STRUCTURE

NIR FRIEDMAN
Computer Science Division, 387 Soda Hall, University of California, Berkeley, CA 94720. nir@cs.berkeley.edu

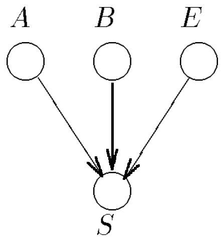
AND

MOISES GOLDSZMIDT
SRI International, 333 Ravenswood Avenue, EK329, Menlo Park, CA 94025. moises@erg.sri.com

Abstract.
We examine a novel addition to the known methods for learning Bayesian networks from data that improves the quality of the learned networks. Our approach explicitly represents and learns the *local structure* in the *conditional probability distributions* (CPDs) that quantify these networks. This increases the space of possible models, enabling the representation of CPDs with a variable number of parameters. The resulting learning procedure induces models that better emulate the interactions present in the data. We describe the theoretical foundations and practical aspects of learning local structures and provide an empirical evaluation of the proposed learning procedure. This evaluation indicates that learning curves characterizing this procedure converge faster, in the number of training instances, than those of the standard procedure, which ignores the local structure of the CPDs. Our results also show that networks learned with local structures tend to be more complex (in terms of arcs), yet require fewer parameters.

1. Introduction

Bayesian networks are graphical representations of probability distributions; they are arguably the representation of choice for uncertainty in artificial intelligence. These networks provide a compact and natural representation, effective inference, and efficient learning. They have been suc-



A	B	E	$\Pr(S \mid A, B, E)$
1	1	1	0.95
1	1	0	0.95
1	0	1	0.20
1	0	0	0.05
0	1	1	0.00
0	1	0	0.00
0	0	1	0.00
0	0	0	0.00

Figure 1. A simple network structure and the associated CPD for variable S (showing the probability values for $S = 1$).

cessfully applied in expert systems, diagnostic engines, and optimal decision making systems.

A Bayesian network consists of two components. The first is a directed acyclic graph (DAG) in which each vertex corresponds to a random variable. This graph describes conditional independence properties of the represented distribution. It captures the *structure* of the probability distribution, and is exploited for efficient inference and decision making. Thus, while Bayesian networks can represent arbitrary probability distributions, they provide computational advantage to those distributions that can be represented with a sparse DAG. The second component is a collection of *conditional probability distributions* (CPDs) that describe the conditional probability of each variable given its parents in the graph. Together, these two components represent a unique probability distribution (Pearl, 1988).

In recent years there has been growing interest in learning Bayesian networks from data; see, for example, Cooper and Herskovits (1992); Buntine (1991b); Heckerman (1995); and Lam and Bacchus (1994). Most of this research has focused on learning the *global* structure of the network, that is, the edges of the DAG. Once this structure is fixed, the parameters in the CPDs quantifying the network are learned by estimating a locally exponential number of parameters from the data. In this article we introduce methods and algorithms for learning *local structures* to represent the CPDs as a part of the process of learning the network. Using these structures, we can model various degrees of complexity in the CPD representations. As we will show, this approach considerably improves the quality of the learned networks.

In its most naive form, a CPD is encoded by means of a tabular representation that is locally exponential in the number of parents of a variable X: for each possible assignment of values to the parents of X, we need to specify a distribution over the values X can take. For example, consider the simple network in Figure 1, where the variables A, B, E and S correspond to the events "alarm armed," "burglary," "earthquake," and "loud alarm sound," respectively. Assuming that all variables are binary, a tabular representation of the CPD for S requires eight parameters, one for each possible state of the parents. One possible quantification of this CPD is given in Figure 1. Note, however, that when the alarm is not armed (i.e., when $A = 0$) the probability of $S = 1$ is zero, regardless of the values B and E. Thus, the interaction between S and its parents is simpler than the eight-way situation that is assumed in the tabular representation of the CPD.

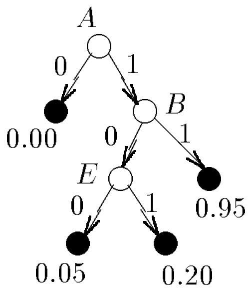
The locally exponential size of the tabular representation of the CPDs is a major problem in learning Bayesian networks. As a general rule, learning many parameters is a liability, since a large number of parameters requires a large training set to be assessed reliably. Thus learning procedures generally encode a bias against structures that involve many parameters. For example, given a training set with instances sampled from the network in Figure 1, the learning procedure might choose a simpler network structure than that of the original network. When the tabular representation is used, the CPD for S requires eight parameters. However, a network with only two parents for S, say A and B, would require only four parameters. Thus, for a small training set, such a network may be preferred, even though it ignores the effect of E on S. This example illustrates that by taking into account the number of parameters, the learning procedure may penalize a large CPD, even if the interactions between the variable and its parents are relatively benign.

Our strategy is to address this problem by explicitly representing the *local structure* of the CPDs. This representation often requires fewer parameters to encode CPDs. This enables the learning procedure to weight each CPD according to the number of parameters it actually requires to capture the interaction between a variable and its parents, rather than the maximal number required by the tabular representation. In other words, this explicit representation of local structure in the network's CPD allows us to adjust the *penalty* incurred by the network to reflect the real complexity of the interactions described by the network.

There are different types of local structures for CPDs, a prominent example is the *noisy-or* gate and its generalizations (Heckerman and Breese, 1994; Pearl, 1988; Srinivas, 1993). In this article, we focus on learning local structures that are motivated by properties of *context-specific* independence (CSI) (Boutilier et al., 1996). These independence statements imply that in some *contexts*, defined by an assignment to variables in the network, the conditional probability of variable X is independent of some of its parents. For example, in the network of Figure 1, when the the alarm is not set

A	B	E	$\Pr(S \vert A, B, E)$
1	1	1	0.95
1	1	0	0.95
1	0	1	0.20
1	0	0	0.05
	*		0.00

(a)



(b)

Figure 2. Two representations of a local CPD structure: (a) a *default table*, and (b) a *decision tree*.

(i.e., the context defined by $A = 0$), the conditional probability does not depend on the value of B and E; $P(S \mid A = 0, B = b, E = e)$ is the same for all values b and e of B and E. As we can see, CSI properties induce equality constraints among the conditional probabilities in the CPDs. In this article, we concentrate on two different representations for capturing the local structure that follows from such equality constraints. These representations, shown in Figure 2, in general require fewer parameters than a tabular representation. Figure 2(a) describes a *default table*, which is similar to the usual tabular representation, except that it does not list all of the possible values of S's parents. Instead, the table provides a *default* probability assignment to all the values of the parents that are not explicitly listed. In this example, the default table requires five parameters instead of the eight parameters required by the tabular representation. Figure 2(b) describes another possible representation based on *decision trees* (Quinlan and Rivest, 1989). Each leaf in the decision tree describes a probability for S, and the internal nodes and arcs encode the necessary information to decide how to choose among leaves, based on the values of S's parents. For example, in the tree of Figure 2(b) the probability of $S = 1$ is 0 when $A = 0$, regardless of the state of B and E; and the probability of $S = 1$ is 0.95 when $A = 1$ and $B = 1$, regardless of the state of E. In this example, the decision tree requires four parameters instead of eight.

Our main hypothesis is that incorporating local structure representations into the learning procedure leads to two important improvements in the quality of the induced models. First, the induced parameters are more reliable. Since these representations usually require less parameters, the frequency estimation for each parameter takes, on average, a larger number of samples into account and thus is more robust. Second, the global structure of the induced network is a better approximation to the real (in)dependencies in the underlying distribution. The use of local structure enables the learning procedure to explore networks that would have incurred an exponential penalty (in terms of the number of parameters required) and thus would have not been taken into consideration. We cannot stress enough the importance of this last point. Finding better estimates of the parameters for a global structure that makes unrealistic independence assumptions will not overcome the deficiencies of the model. Thus, it is crucial to obtain a good approximation of the global structure.

The experiments described in Section 5 confirm our main hypothesis. Moreover, the results in that section show that the use of local representations for the CPDs significantly affects the learning process itself: The learning procedures require fewer data samples in order to induce a network that better approximates the target distribution.

The main contributions of this article are: the derivation of the scoring functions and algorithms for learning the local representations; the formulation of the hypothesis introduced above, which uncovers the benefits of having an explicit local representation for CPDs; and the empirical investigation that validates this hypothesis.

CPDs with local structure have often been used and exploited in tasks of knowledge acquisition from experts; as we already mentioned above, the *noisy-or* gate and its generalizations are well known examples (Heckerman and Breese, 1994; Pearl, 1988; Srinivas, 1993). In the context of learning, several authors have noted that CPDs can be represented via logistic regression, noisy-or, and neural networks (Buntine, 1991b; Diez, 1993; Musick, 1994; Neal, 1992; Spiegelhalter and Lauritzen, 1990). With the exception of Buntine, these authors have focused on the case where the network structure is fixed in advance, and motivate the use of local structure for learning reliable parameters. The method proposed by Buntine (1991b) is not limited to the case of a fixed structure; he also points to the use of decision trees for representing CPDs. Yet, in that paper, he does not provide empirical or theoretical evidence for the benefits of using local structured representations with regards to a more accurate induction of the global structure of the network. To the best of our knowledge, the benefits that relate to that, as well as to the convergence speed of the learning procedure (in terms of the number of training instances), have been unknown in the literature prior to our work.

The reminder of this article is organized as follows: In Section 2 we review the definition of Bayesian networks, and the scores used for learning these networks. In Section 3 we describe the two forms of local structured CPDs we consider in this article. In Section 4 we formally derive the score for learning networks with CPDs represented as default tables and decision trees, and describe the procedures for learning these structures. In Section 5 we describe the experimental results. We present our conclusions in Section 6.

2. Learning Bayesian Networks

Consider a finite set $\mathbf{U} = \{X_1, \ldots, X_n\}$ of discrete random variables where each variable X_i may take on values from a finite domain. We use capital letters such as X, Y, Z to denote variable names, and lowercase letters such as x, y, z to denote specific values taken by those variables. The set of values X can attain is denoted as $Val(X)$; the cardinality of this set is denoted as $||X|| = |Val(X)|$. Sets of variables are denoted by boldface capital letters such as $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$, and assignments of values to the variables in these sets are denoted by boldface lowercase letters such as $\mathbf{x}, \mathbf{y}, \mathbf{z}$ (we use $Val(\mathbf{X})$ and $||\mathbf{X}||$ in the obvious way).

Let P be a joint probability distribution over the variables in $\mathbf{U}$, and let $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ be subsets of $\mathbf{U}$. $\mathbf{X}$ and $\mathbf{Y}$ are *conditionally independent*, given $\mathbf{Z}$, if for all $\mathbf{x} \in Val(\mathbf{X}), \mathbf{y} \in Val(\mathbf{Y})$, and $\mathbf{z} \in Val(\mathbf{Z})$, we have that $P(\mathbf{x} \mid \mathbf{z}, \mathbf{y}) = P(\mathbf{x} \mid \mathbf{z})$ whenever $P(\mathbf{y}, \mathbf{z}) > 0$.

A *Bayesian network* is an annotated DAG that encodes a joint probability distribution of a domain composed of a set of random variables. Formally, a Bayesian network for $\mathbf{U}$ is the pair $B = \langle G, \mathcal{L} \rangle$. G is a DAG whose nodes correspond to the random variables $X_1, \ldots, X_n$, and whose edges represent direct dependencies between the variables. The graph structure G encodes the following set of independence statements: each variable X_i is independent of its nondescendants, given its parents in G. The set composed of a variable and its parents is usually referred to as a *family*.

Standard arguments (Pearl, 1988) show that any distribution P that satisfies the independence statements encoded in the graph G can be *factored* as

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid \mathbf{Pa}_i), \tag{1}$$

where $\mathbf{Pa}_i$ denote the parents of X_i in G. Note that to completely specify a distribution of this form, we only need to provide the conditional probabilities on the right hand side. This is precisely the second component of the Bayesian network, namely $\mathcal{L}$. This set of CPDs specify the conditional probability $P(X_i \mid \mathbf{Pa}_i)$ for all variables X_i. It immediately follows that there is exactly one distribution that has the form of Equation 1 with the conditional probabilities specified in $\mathcal{L}$.

When we deal with discrete variables, we usually represent the CPDs in $\mathcal{L}$ as conditional probability tables such as the one in Figure 1. These tables contain a parameter $\theta_{x_i|\mathbf{pa}_i}$ for each value $x_i \in \mathit{Val}(X_i)$ and $\mathbf{pa}_i \in \mathit{Val}(\mathbf{Pa}_i)$.

The problem of learning a Bayesian network can be now stated as follows. Given a *training set* $D = \{\mathbf{u}_1, \ldots, \mathbf{u}_N\}$ of instances of $\mathbf{U}$, find a network $B = \langle G, \mathcal{L} \rangle$ that *best matches* D.[1] To formalize the notion of goodness of fit of a network with respect to the data, we normally introduce a scoring function, and to solve the optimization problem we usually rely on heuristic search techniques over the space of possible networks (Heckerman, 1995). Several different scoring functions have been proposed in the literature. In this article we focus our attention on the ones that are most frequently used: the *Minimal Description Length* (MDL) score (Lam and Bacchus, 1994) and the BDe score (Heckerman et al., 1995a).

2.1. THE MDL SCORE

The MDL principle (Rissanen, 1989) is motivated by *universal coding*. Suppose that we are given a set D of instances, which we would like to store in our records. Naturally, we would like to conserve space and save a compressed version of D. One way of compressing the data is to find a suitable model for D that the encoder can use to produce a compact version of D. Moreover, as we want to be able to recover D, we must also store the model used by the encoder to compress D. The *total description length* is then defined as the sum of the length of the compressed version of D and the length of the description of the model used in the compression. The MDL principle dictates that the optimal model is the one (from a particular class of interest) that minimizes the total description length.

In the context of learning Bayesian networks, the model is a network. Such a network, B, describes a probability distribution, P_B, over the instances appearing in the data. Using this distribution, we can build an encoding scheme that assigns shorter code words to more probable instances (e.g., using Shannon encoding or a Huffman code; see Cover and Thomas (1991)). According to the MDL principle, we should choose a network, B, such that the combined length of the network description and the encoded data (with respect to P_B) is minimized. This implies that the learning procedure balances the complexity of the induced network with the degree of accuracy with which the network represents the frequencies in D.

We now describe in detail the representation length required for the storage of both the network and the coded data. The MDL score of a candidate network is defined as the total description length. To store a

[1] Throughout this article we will assume that the training data is *complete*, i.e., that each $\mathbf{u}_i$ assigns values to all variables in $\mathbf{U}$. Existing solutions to the problem of missing values apply to the approaches we discuss below; see Heckerman (1995).

network $B = \langle G, \mathcal{L} \rangle$, we need to describe $\mathbf{U}$, G, and $\mathcal{L}$.

To describe $\mathbf{U}$, we store the the number of variables, n, and the cardinality of each variable X_i. Since $\mathbf{U}$ is the same for all candidate networks, we can ignore the description length of $\mathbf{U}$ in the comparisons between networks.

To describe the DAG G, it is sufficient to store for each variable X_i a description of $\mathbf{Pa}_i$ (namely, its parents in G). This description consists of the number of parents, k, followed by the index of the set $\mathbf{Pa}_i$ in some (agreed upon) enumeration of all $\binom{n}{k}$ sets of this cardinality. Since we can encode the number k using $\log n$ bits, and we can encode the index using $\log \binom{n}{k}$ bits, the description length of the graph structure is[2]

$$DL_{graph}(G) = \sum_i \left(\log n + \log \binom{n}{|\mathbf{Pa}_i|} \right).$$

To describe the CPDs in $\mathcal{L}$, we must store the parameters in each conditional probability table. For the table associated with X_i, we need to store $||\mathbf{Pa}_i||(||X_i|| - 1)$ parameters. The representation length of these parameters depends on the number of bits we use for each numeric parameter. The usual choice in the literature is $1/2 \log N$ (see Friedman and Yakhini (1996) for a thorough discussion of this point). Thus, the encoding length of X_i's CPD is

$$DL_{tab}(X_i, \mathbf{Pa}_i) = \frac{1}{2}||\mathbf{Pa}_i||(||X_i|| - 1) \log N.$$

To encode the training data, we use the probability measure defined by the network B to construct a Huffman code for the instances in D. In this code, the exact length of each codeword depends on the probability assigned to that particular instance. There is no closed-form description of this length. However, it is known (Cover and Thomas, 1991) that we can approximate the optimal encoding length using $-\log P_B(\mathbf{u})$ as the encoding length of each instance $\mathbf{u}$. Thus, the description length of the data is approximated by

$$DL_{data}(D \mid B) = -\sum_{i=1}^{N} \log P_B(\mathbf{u}_i).$$

We can rewrite this expression in a more convenient form. We start by introducing some notation. Let $\hat{P}_D$ be the empirical probability measure

[2]Since description lengths are measured in terms of bits, we use logarithms of base 2 throughout this article.

induced by the data set D. More precisely, we define

$$\hat{P}_D(A) = \frac{1}{N}\sum_{i=1}^{N} 1_A(\mathbf{u}_i) \text{ where } 1_A(\mathbf{u}) = \begin{cases} 1 & \text{if } \mathbf{u} \in A \\ 0 & \text{if } \mathbf{u} \notin A \end{cases}$$

for all events of interest, i.e., $A \subseteq \mathit{Val}(\mathbf{U})$. Let $N^D_{\mathbf{X}}(\mathbf{x})$ be the number of instances in D where $\mathbf{X} = \mathbf{x}$ (from now on, we omit the subscript from $\hat{P}_D$, and the superscript and the subscript from $N^D_{\mathbf{X}}$, whenever they are clear from the context). Clearly, $N(\mathbf{x}) = N \cdot \hat{P}(\mathbf{x})$. We use Equation 1 to rewrite the representation length of the data as

$$\begin{aligned} DL_{data}(D \mid B) &= -\sum_{j=1}^{N} \log P_B(\mathbf{u}_j) \\ &= -N \sum_{u} \hat{P}(u) \log \prod_i P(x_i \mid \mathbf{pa}_i) \\ &= -\sum_i \sum_{x_i, \mathbf{pa}_i} N(x_i, \mathbf{pa}_i) \log P(x_i \mid \mathbf{pa}_i). \end{aligned} \tag{2}$$

Thus, the encoding of the data can be decomposed as a sum of terms that are "local" to each CPD: these terms depend only on the counts $N(x_i, \mathbf{pa}_i)$. Standard arguments show the following.

Proposition 2.1: *If $P(X_i \mid \mathbf{Pa}_i)$ is represented as a table, then the parameter values that minimize $DL_{data}(D \mid B)$ are $\theta_{x_i|\mathbf{pa}_i} = \hat{P}(x_i \mid \mathbf{pa}_i)$.*

Thus, given a fixed network structure G, learning the parameters that minimize the description length is straightforward: we simply compute the appropriate long-run fractions from the data.

Assuming that we assign parameters in the manner prescribed by this proposition, we can rewrite $DL_{data}(D \mid B)$ in a more convenient way in terms of conditional entropy: $N \sum_i H(X_i \mid \mathbf{Pa}_i)$, where $H(X \mid Y) = -\sum_{x,y} \hat{P}(x,y) \log \hat{P}(x \mid y)$ is the *conditional entropy* of X, given Y. This formula provides an information-theoretic interpretation to the representation of the data: it measures how many bits are necessary to encode the value of X_i, once we know the value of $\mathbf{Pa}_i$.

Finally, the MDL score of a candidate network structure G, assuming that we choose parameters as prescribed above, is defined as the total description length

$$\begin{aligned} DL(G, D) &= DL_{graph}(G) + \sum_i DL_{tab}(X_i, \mathbf{Pa}_i) + \\ & \quad N \sum_i H(X_i \mid \mathbf{Pa}_i). \end{aligned} \tag{3}$$

According to the MDL principle, we should strive to find the network structure that minimizes this description length. In practice, this is usually done by searching over the space of possible networks.

2.2. THE BDE SCORE

Scores for learning Bayesian networks can also be derived from methods of Bayesian statistics. A prime example of such scores is the BDe score, proposed by Heckerman et al. (1995a). This score is based on earlier work by Cooper and Herskovits (1992) and Buntine (1991b). The BDe score is (proportional to) the posterior probability of each network structure, given the data. Learning amounts to searching for the network(s) that maximize this probability.

Let G^h denote the hypothesis that the underlying distribution satisfies the independencies encoded in G (see Heckerman et al. (1995a) for a more elaborate discussion of this hypothesis). For a given structure G, let Θ_G represent the vector of parameters for the CPDs quantifying G. The posterior probability we are interested in is $\Pr(G^h \mid D)$. Using Bayes' rule we write this term as

$$\Pr(G^h \mid D) = \alpha \Pr(D \mid G^h) \Pr(G^h), \tag{4}$$

where α is a normalization constant that does not depend on the choice of G. The term $\Pr(G^h)$ is the prior probability of the network structure, and the term $\Pr(D \mid G^h)$ is the probability of the data, given that the network structure is G.

There are several ways of choosing a prior over network structures. Heckerman et al. suggest choosing a prior $\Pr(G^h) \propto a^{\Delta(G,G')}$, where $\Delta(G, G')$, is the difference in edges between G and a *prior* network structure G', and $0 < a < 1$ is penalty for each such edge. In this article, we use a prior based on the MDL encoding of G. We let $\Pr(G^h) \propto 2^{DL_{graph}(G)}$.

To evaluate the $\Pr(D \mid G^h)$ we must consider all possible parameter assignments to G. Thus,

$$\Pr(D \mid G^h) = \int \Pr(D \mid \Theta_G, G^h) \Pr(\Theta_G \mid G^h) d\Theta_G, \tag{5}$$

where $\Pr(D \mid \Theta_G, G^h)$ is defined by Equation 1, and $\Pr(\Theta_G \mid G^h)$ is the prior density over parameter assignments to G. Heckerman et al. (following Cooper and Herskovits (1992)) identify a set of assumptions that justify decomposing this integral. Roughly speaking, they assume that each distribution $P(X_i \mid \mathbf{pa}_i)$ can be learned independently of all other distributions.

Using this assumption, they rewrite $\Pr(D \mid G^h)$ as

$$\Pr(D \mid G^h) = \prod_i \prod_{\mathbf{pa}_i} \int \prod_{x_i} \theta_{x_i|\mathbf{pa}_i}^{N(x_i,\mathbf{pa}_i)} \Pr(\Theta_{X_i|\mathbf{pa}_i} \mid G^h) d\Theta_{X_i|\mathbf{pa}_i}. \tag{6}$$

(This decomposition is analogous to the decomposition in Equation 2.) When the prior on each multinomial distribution $\Theta_{X_i|\mathbf{pa}_i}$ is a *Dirichlet prior*, the integrals in Equation 6 have a closed-form solution (Heckerman, 1995).

We briefly review the properties of Dirichlet priors. For more detailed description, we refer the reader to DeGroot (1970). A Dirichlet prior for a multinomial distribution of a variable X is specified by a set of *hyperparameters* $\{N'_x : x \in \mathit{Val}(X)\}$. We say that

$$\Pr(\Theta_X) \sim \mathit{Dirichlet}(\{N'_x : x \in \mathit{Val}(X)\})$$

if

$$\Pr(\Theta_X) = \alpha \prod_x \theta_x^{N'_x},$$

where α is a normalization constant. If the prior is a Dirichlet prior, the probability of observing a sequence of values of X with counts $N(x)$ is

$$\int \prod_x \theta_x^{N(x)} \Pr(\Theta_X \mid G^h) d\Theta_X = \frac{\Gamma(\sum_x N'_x)}{\Gamma(\sum_x (N'_x + N(x)))} \prod_x \frac{\Gamma(N'_x + N(x))}{\Gamma(N'_x)},$$

where $\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt$ is the *Gamma* function that satisfies the properties $\Gamma(1) = 1$ and $\Gamma(x+1) = x\Gamma(x)$.

Returning to the BDe score, if we assign to each $\Theta_{X_i|\mathbf{pa}_i}$ a Dirichlet prior with hyperparameters $N'_{x_i|\mathbf{pa}_i}$, then

$$\Pr(D \mid G^h) = \prod_i \prod_{\mathbf{pa}_i} \frac{\Gamma(\sum_{x_i} N'_{x_i|\mathbf{pa}_i})}{\Gamma(\sum_{x_i} N'_{x_i|\mathbf{pa}_i} + N(\mathbf{pa}_i))} \prod_{x_i} \frac{\Gamma(N'_{x_i|\mathbf{pa}_i} + N(x_i, \mathbf{pa}_i))}{\Gamma(N'_{x_i|\mathbf{pa}_i})}. \tag{7}$$

There still remains a problem with the direct application of this method. For each possible network structure we would have to assign priors on the parameter values. This is clearly infeasible, since the number of possible structures is extremely large. Heckerman et al. propose a set of assumptions that justify a method by which, given a prior network B^p and an equivalent sample size N', we can assign prior probabilities to parameters in every possible network structure. The prior assigned to $\Theta_{X_i|\mathbf{pa}_i}$ in a structure G is computed from the prior distribution represented in B^p. In this method, we assign $N'_{x_i|\mathbf{pa}_i} = N' \cdot P_{B^p}(x_i, \mathbf{pa}_i)$. (Note that $\mathbf{Pa}_i$ are the parents of

X_i in G, but not necessarily in B^p.) Thus, their proposal essentially uses the conditional probability of X_i, given $\mathbf{pa}_i$, in the prior network B^p as the expected probability. Similarly, the confidence in the prior (e.g., the magnitude of the hyperparameters) is proportional to the expected number of occurrences of the values of $\mathbf{Pa}_i$.

The exposition above shows how to score a network structure G. In order to make predictions about the probability distribution over the set of variables $\mathbf{u}$, given a structure G, we need to compute the set of parameters to quantify the G. According to the Bayesian methodology, we should average over all possible assignments to Θ_G. Thus,

$$\Pr(\mathbf{u} \mid D, G^h) = \int \Pr(\mathbf{u} \mid \Theta_G, G^h) \Pr(\Theta_G \mid D, G^h) d\Theta_G.$$

Once again we can decompose this term using the structure of the G. Using the assumptions stated above of completely observable data, and parameter independence, we get that $\Pr(\mathbf{u} \mid D, G^h) = \prod_i \Pr(x_i \mid \mathbf{pa}_i, D, G^h)$ where

$$\Pr(x_i \mid \mathbf{pa}_i, D, G^h) = \int \theta_{X_i|\mathbf{pa}_i} \Pr(\theta_{X_i|\mathbf{pa}_i} \mid \mathbf{pa}_i, D, G^h) d\theta_{X_i|\mathbf{pa}_i}.$$

If we use Dirichlet priors, then these integrals have the closed-form solution

$$\Pr(x_i \mid \mathbf{pa}_i, D, G^h) = \frac{N'_{x_i|\mathbf{pa}_i} + N(x_i, \mathbf{pa}_i)}{\sum_{x_i} N'_{x_i|\mathbf{pa}_i} + N(\mathbf{pa}_i)}.$$

When we consider large data sets, the MDL score and the BDe score tend to score candidate structures similarly. More precisely, these two scores are asymptotically equivalent. This equivalence can be derived by using asymptotic approximations to the $\Gamma(\cdot)$ function in Equation 7, as done by Bouckaert (1994), or by using a general result of Schwarz (1978). Schwarz shows that, given some regularity constraints on the prior,

$$log \Pr(D \mid G^h) \approx log \Pr(D \mid \hat{\Theta}_G, G^h) - \frac{d}{2} \log N, \tag{8}$$

where $\hat{\Theta}_G$ are the maximum likelihood parameters for G, given D, and d is the *dimension* of G, which in our setting is the number of free parameters in G.

Note that the term on the right-hand side of Equation 8 (which one attempts to maximize) is the negative of the MDL score of Equation 3 (which one attempts to minimize), when we ignore the description of G, which corresponds to the logarithm of the prior $\Pr(G^h)$. Note also that this term is negligible in the asymptotic analysis, since it does not depend on N.

3. Local Structure

In the discussion above, we have assumed the standard tabular representation of the CPDs quantifying the networks. This representation requires that for each variable X_i we encode a locally exponential number, $||\mathbf{Pa}_i||(||X_i|| - 1)$, of parameters. In practice, however, the interaction between X_i and its parents $\mathbf{Pa}_i$ can be more benign, and some regularities can be exploited to represent the same information with fewer parameters. In the example of Figure 1, the the CPD for S can be encoded with four parameters, by means of the decision tree of Figure 2(b), in contrast to the eight parameters required by the tabular representation.

A formal foundation for representing and reasoning with such regularities is provided in the notion of *context-specific independence* (CSI) (Boutilier et al., 1996). Formally, we say that $\mathbf{X}$ and $\mathbf{Y}$ are *contextually independent*, given $\mathbf{Z}$ and the *context* $\mathbf{c} \in \mathit{Val}(\mathbf{C})$, if

$$P(\mathbf{X} \mid \mathbf{Z}, \mathbf{c}, \mathbf{Y}) = P(\mathbf{X} \mid \mathbf{Z}, \mathbf{c}) \text{ whenever } P(\mathbf{Y}, \mathbf{Z}, \mathbf{c}) > 0. \tag{9}$$

CSI statements are more specific than the conditional independence statements captured by the Bayesian network structure. CSI implies the independence of X and Y, given a specific value of the context variable(s), while conditional independence applies for all value assignments to the conditioning variable. As shown by Boutilier et al. (1996), the representation of CSI leads to several benefits in knowledge elicitation, compact representation, and computational efficiency. As we show here, CSI is also beneficial to learning, since models can be quantified with fewer parameters.

As we can see from Equation 9, CSI statements force equivalence relations between certain conditional probabilities. If X_i is contextually independent of $\mathbf{Y}$ given $\mathbf{Z}$ and $\mathbf{c} \in \mathit{Val}(\mathbf{C})$, then $P(X_i \mid \mathbf{Z}, \mathbf{c}, \mathbf{y}) = P(X_i \mid \mathbf{Z}, \mathbf{c}, \mathbf{y}')$ for $\mathbf{y}, \mathbf{y}' \in \mathit{Val}(\mathbf{Y})$. Thus, if the parent set $\mathbf{Pa}_i$ of X_i is equal to $\mathbf{Y} \cup \mathbf{Z} \cup \mathbf{C}$, such CSI statements will induce equality constraints among the conditional probability of X_i given its parents. This observation suggests an alternative way of thinking of local structure in terms of the partitions they induce on the possible values of the parents of each variable X_i. We note that while CSI properties imply such partitions, not all partitions can be characterized by CSI properties.

These partitions impose a structure over the CPDs for each X_i. In this article we are interested in representations that explicitly capture this structure that reduces the number of parameters to be estimated by the learning procedure. We focus on two representations that are relatively straightforward to learn. The first one, called *default tables*, represent a set of singleton partitions with one additional partition that can contain several values of $\mathbf{Pa}_i$. Thus, the savings it will introduce depends on how many values in

$Val(\mathbf{Pa}_i)$ can be grouped together. The second representation is based on decision trees and can represent more complex partitions. Consequently it can reduce the number of parameters even further. Yet the induction algorithm for decision trees is somewhat more complex than that of default tables.

We introduce a notation that simplifies the presentation below. Let L be a representation for the CPD for X_i. We capture the partition structure represented by L using a *characteristic* random variable Υ_L. This random variable maps each value of $\mathbf{Pa}_i$ to the partition that contains it. Formally, $\Upsilon_L(\mathbf{pa}_i) = \Upsilon_L(\mathbf{pa}'_i)$ for two values $\mathbf{pa}_i$ and $\mathbf{pa}'_i$ of $\mathbf{Pa}_i$, if and only if these two values are in the same partition in L. It is easy to see that from the definition of Υ_L, we get $\mathcal{P}(X_i \mid \Upsilon_L) = \mathcal{P}(X_i \mid \mathbf{Pa}_i)$, since if $\mathbf{pa}_i$ and $\mathbf{pa}'_i$ are in the same partition, it must be that $P(X_i \mid \mathbf{pa}_i) = P(X_i \mid \mathbf{pa}'_i)$. This means that we can describe the parameterization of the structure L in terms of the characteristic random variable Υ_L as follows: $\Theta_L = \{\theta_{x_i|\upsilon} : x_i \in Val(X_i), \upsilon \in Val(\Upsilon_L)\}$.

As an example, consider the tabular CPD representation, in which no CSI properties are taken into consideration. This implies that the corresponing partitions contain exactly one value for $\mathbf{Pa}_i$. Thus, in this case, $Val(\Upsilon_L)$ is isomorphic to $Val(\mathbf{Pa}_i)$. CPD representations that specify CSI relations will have fewer partitions, and thus will require fewer parameters.

In the sections below we formally describe default tables and decision trees, and the partition structures they represent.

3.1. DEFAULT TABLES

A default table is similar to a standard tabular representation of a CPD, except that only a subset of the possible values of the parents of a variable are explicitly represented as rows in the table. The values of the parents that are not explicitly represented as individual rows are mapped to a special row called the *default row*. The underlying idea is that the probability of a node X is the same for all the values of the parents that are mapped to the default row; therefore, there is no need to represent these values separately in several entries. Consequently, the number of parameters explicitly represented in a default table can be smaller than the number of parameters in a tabular representation of a CPD. In the example showing in Figure 2(a), all the values of the parents of S, where $A = 0$ (the alarm is not armed), are mapped to the default row in the table, since the probability of $S = 1$ is the same in all of these situations, regardless of the values of B and E.

Formally, a default table is an object $\mathcal{D} = (S_{\mathcal{D}}, \Theta_{\mathcal{D}})$. $S_{\mathcal{D}}$ describes the *structure* of the table, namely, which *rows* are represented explicitly, and which are represented via the default row. We define $Rows(\mathcal{D}) \subseteq Val(\mathbf{Pa}_i)$

to be the set of rows in $\mathcal{D}$ that are represented explicitly. This structure defines the following characteristic random variable. If $\mathbf{pa}_i \in Rows(\mathcal{D})$, then the value $\Upsilon(\mathbf{pa}_i)$ is the partition that contains only $\mathbf{pa}_i$. If $\mathbf{pa}_i \notin Rows(\mathcal{D})$, then the value $\Upsilon(\mathbf{pa}_i)$ is the default partition that contains all the values that are not explicitly represented, that is $Val(\mathbf{Pa}_i) - Rows(\mathcal{D})$. Thus, the partitions defined by the default table correspond to the rows in the explicit representation of the table (e.g., as in Figure 2(a)).

The set of parameters, $\Theta_{\mathcal{D}}$, constitutes the parameterization for $\mathcal{D}$. It contains parameters $\theta_{x_i|v}$ for each value $x_i \in Val(\Upsilon_{\mathcal{D}})$. To determine $P(x_i \mid \mathbf{pa}_i)$ from this representation we need to consider two cases. If $\mathbf{pa}_i \in Rows(\mathcal{D})$, then $P(x_i \mid \mathbf{pa}_i) = \theta_{x_i|\Upsilon_{\mathcal{D}}=\{\mathbf{pa}_i\}}$. If $\mathbf{pa}_i \notin Rows(\mathcal{D})$, then $P(x_i \mid \mathbf{pa}_i) = \theta_{x_i|\Upsilon_{\mathcal{D}}=D}$, where $D = Val(\mathbf{Pa}_i) = Rows(\mathcal{D})$ is the partition that corresponds to the default row.

3.2. DECISION TREES

A *decision tree* for variable X is a tree in which each internal node is annotated with a parent variable, outgoing edges from a particular node are annotated with the values that the variable represented by that node can take, and leaves are annotated with a probability distribution over X. The process of retrieving the probability of X, given a value of its parents, is as follows. We start at the root node and traverse the tree until we reach a leaf. At each internal node, we choose which subtree to traverse by testing the value of the parent that annotates that node and following the outgoing edge that corresponds to that value. Thus, suppose that we would like to know $\Pr(S = 1 \mid A = 1, B = 0, E = 1)$ in the tree shown in Figure 2(b). We follow the edge to the right subtree at A, since this edge is annotated with the value 1 for A. Similarly, we follow the left edge at B (annotated with 0), and again the right edge at E, till we reach the appropriate leaf.

Formally, we denote a tree as an object $\mathcal{T} = (S_{\mathcal{T}}, \Theta_{\mathcal{T}})$. The first component, $S_{\mathcal{T}}$ represents the structure of the tree, and is defined recursively. A tree can be either a leaf or a composite tree. A leaf is represented by a structure equal to a special constant $S_{\mathcal{T}} = \Lambda$. A composite tree is represented by a structure of the form $S_{\mathcal{T}} = \langle Y, \{S_{\mathcal{T}_y} : y \in Val(Y)\}\rangle$, where Y is the test variable at the root of the tree, and $S_{\mathcal{T}_y}$ is a tree structure, for each value y of Y. We denote by $Label(\mathcal{T})$ the variable tested at the root of $\mathcal{T}$, and by $Sub(\mathcal{T}, v)$ the subtree associated with the value v of $Label(\mathcal{T})$.

Finally, we need to describe the partitions induced by this representation. Let a *path* be the set of arcs lying between the root and a leaf. A path is *consistent* with $\mathbf{pa}_i$ if the labeling of the path is consistent with the assignment of values in $\mathbf{pa}_i$. It is easy to verify that for every $\mathbf{pa}_i \in Val(\mathbf{Pa}_i)$ there is a unique consistent path in the tree. The partitions induced by a decision tree correspond to the set of paths in the tree, where the partition that correspond to a particular path p consists of all the value assignments that p is consistent with. Again, we define the set of parameters, $\Theta_{\mathcal{T}}$, to contain parameters $\theta_{x_i|\upsilon}$ for each value $x_i \in \mathit{Val}(\Upsilon_{\mathcal{D}})$. That is, we associate with each (realizable) path in the tree a distribution over X_i. To determine $P(x_i \mid \mathbf{pa}_i)$ from this representation, we simply choose $\theta_{x_i|\upsilon}$, where $\upsilon = \{\mathbf{pa}'_i \mid \mathbf{pa}'_i$ is consistent with $p\}$, where p is the (unique) path that is consistent with $\mathbf{pa}_i$.

4. Learning Local Structure

We start this section by deriving both the MDL and BDe scoring functions for default table and decision tree representations. We then describe the procedures for searching for high scoring networks. Note that the material in this section can be easily generalized to derive a score and produce a learning procedure for any structured representation of the CPDs that represents a partition over the values of the parent variables. (See Boutilier et al. (1996) for a discussion of such representations.)

4.1. SCORING FUNCTIONS

We introduce some notations necessary for our derivations. Let L denote a local representation of $P(X_i \mid \mathbf{Pa}_i)$, e.g., a default table, a tree, or a (complete) table. We denote by S_L the structure of the local representation L, and by Θ_L, the parameterization of L. We assume that $\Theta_L = \{\theta_{x_i|\upsilon} : x_i \in \mathit{Val}(X_i), \upsilon \in \mathit{Val}(\Upsilon_L)\}$.

4.1.1. *MDL Score for Local Structure*

Let $B = \langle G, \{L_i\}\rangle$ be a Bayesian network, where L_i is the local representation of $P(X_i \mid \mathbf{Pa}_i)$. The MDL encoding of the DAG G remains the same as in Section 2.1. Changes occur in the encoding of L_i. We now have to encode the structure S_{L_i} and the parameters Θ_{L_i}. Additionally, the choice of optimal parameters, given the data, now depends on the choice of local structure.

First, we describe the encoding of S_L for both default table and tree representations.

When L is a default table $\mathcal{D}$, we need to describe the set of rows that are represented explicitly in the table, that is, $\mathit{Rows}(\mathcal{D})$. We start by encoding the number $k = |\mathit{Rows}(\mathcal{D})|$; then we describe $\mathit{Rows}(\mathcal{D})$ by encoding its index in some (agreed upon) enumeration of all $\binom{||\mathbf{Pa}_i||}{k}$ sets of this cardinality.

Thus, the description length of the structure $\mathcal{D}$ is

$$DL_{local\text{-}struct}(\mathcal{D}) = \log ||\mathbf{Pa}_i|| + \log \binom{||\mathbf{Pa}_i||}{k}.$$

When L is a tree $\mathcal{T}$, we need to encode the structure of the tree, and the labeling of internal nodes in the tree. We use the encoding proposed by Quinlan and Rivest (1989).[3] A tree is encoded recursively as follows: a leaf is encoded by a single bit with value equal to 0. The encoding of a composite tree starts with a bit set to the value 1, to differentiate it from a leaf, followed by a description of the associated test variable and the description of all the immediate subtrees. The encoding of the test variable depends on the position of the node in the tree. At the root, the test variable can be any of X_i's parents. In contrast, at the a subtree, the choice of a test variable is more restricted, since along a single path we test each variable at most once. In general, if there are k variables that have not been tested yet in the path from the root to the current node in the tree, then we need to store only $\log(k)$ bits to describe the test variable. The total description length of the tree structure is described by the following recurring formula:

$$DL_{\mathcal{T}}(\mathcal{T}, k) = \begin{cases} 1 & \text{if } \mathcal{T} \text{ is a leaf,} \\ 1 + \log(k) + \sum_i DL_{\mathcal{T}}(\mathcal{T}_i, k-1) & \text{if } \mathcal{T} \text{ is a composite tree} \\ & \text{with subtrees } \mathcal{T}_1, \ldots, \mathcal{T}_m. \end{cases}$$

Using this formula, we define $DL_{local\text{-}struct}(\mathcal{T}) = DL_{\mathcal{T}}(\mathcal{T}, |\mathbf{Pa}_i|)$.

Next, we encode the $(||X_i|| - 1)||\Upsilon_L||$ parameters for L with description length

$$DL_{param}(L) = \frac{1}{2}(||X_i|| - 1)||\Upsilon_L|| \log N.$$

Finally, as we did in Section 2.1, we describe the encoding of the data given the model using Equation 2.

We now generalize Proposition 2.1 to describe the optimal choice of parameters for a network when CPDs are represented using local structure.

Proposition 4.1: *If $P(X_i \mid \mathbf{Pa}_i)$ is represented by local representation L_i, for $i = 1, \ldots, n$, then we can rewrite $DL_{data}(D \mid B)$ as*

$$DL_{data}(D \mid B) = -N \sum_i \sum_{v \in Val(\Upsilon_{L_i})} \sum_{x_i} \hat{P}(x_i, \Upsilon_{L_i} = v) \log \theta_{x_i|v}.$$

[3] Wallace and Patrick (1993) note that this encoding is inefficient, in the sense that the number of legal tree structures that can be described by n-bit strings, is significantly smaller than 2^n. Their encoding, which is more efficient, can be easily incorporated into our MDL encoding. For clarity of presentation, we use the Quinlan and Rivest encoding in this article.

Moreover, the parameter values for L that minimize $DL_{data}(D \mid B)$ are

$$\theta_{x_i|\Upsilon_{L_i}=\upsilon} = \hat{P}(x_i \mid \Upsilon_{L_i} = \upsilon).$$

As in the case of tabular CPD representation, DL_{data} is minimized when the parameters correspond to the appropriate frequencies in the training data. As a consequence of this result, we find that for a fixed local structure L, the minimal representation length of the data is simply $N \cdot H(X \mid \Upsilon_L)$. Thus, once again we derive an information-theoretic interpretation of $DL_{data}(\Theta_L, D)$. This interpretation shows that the encoding of X depends only on the values of Υ_L. From the *data processing inequality* (Cover and Thomas, 1991) it follows that $H(X_i \mid \Upsilon_L) \geq H(X_i \mid \mathbf{Pa}_i)$. This implies that a local structure cannot fit the data better than a tabular CPD. Nevertheless, as our experiments confirm, the reduction in the number of parameters can compensate for the potential loss in information.

To summarize, the MDL score for a graph structure augmented with a local structure L_i for each X_i is

$$\begin{aligned} DL(G, L_1, \ldots, L_n, D) &= DL_{graph}(G) + \sum_i (DL_{local\text{-}struct}(L_i) + DL_{param}(L_i)) \\ &\quad + N \sum_i H(X \mid \Upsilon_{L_i}). \end{aligned}$$

4.1.2. *BDe Score for Local Structure*

We now describe how to extend the BDe score for learning local structure. Given the hypothesis G^h, we denote by $\mathcal{L}_G^h$ the hypothesis that the underlying distribution satisfies the constraints of a set of local structures $\mathcal{L} = \{L_i : 1 \leq i \leq n\}$, where L_i is a local structure for the CPD of X_i in G.

Using Bayes' rule, it follows that

$$\Pr(G^h, L_G^h \mid D) \propto \Pr(D \mid L_G^h, G^h) \Pr(L_G^h \mid G^h) \Pr(G^h).$$

The specification of priors on local structures presents no additional complications other than the specification of priors for the structure of the network G^h. Buntine (1991a, 1993), for example, suggests several possible priors on decision trees. A natural prior over local structures is defined via the MDL description length, by setting $\Pr(\mathcal{L}_G^h \mid G^h) \propto 2^{-\sum_i DL_{local\text{-}struct}(L_i)}$.

For the term $\Pr(D \mid L_G^h, G^h)$, we make an assumption of parameter independence, similar to the one made by Heckerman et al. (1995a) and by Buntine (1991b): the parameter values for each possible value of the characteristic variable Υ_{L_i} are independent of each other. Thus, each multinomial sample is independent of the others, and we can derive the analogue of Equation 6:

$$\Pr(D \mid \mathcal{L}_G^h, G^h) = \prod_i \prod_{\upsilon \in \mathit{Val}(\Upsilon_{L_i})} \int \prod_{x_i} \theta_{x_i|\upsilon}^{N(x_i,\upsilon)} \Pr(\Theta_{X_i|\upsilon} \mid {L_i}^h, G^h) d\Theta_{X_i|\upsilon} \tag{10}$$

(This decomposition is analogous to the one described in Proposition 4.1.) As before, we assume that the priors $\Pr(\Theta_{X_i|\upsilon} \mid \mathcal{L}_G^h, G^h)$ are Dirichlet, and thus we get a closed-form solution for Equation 10,

$$\Pr(D \mid \mathcal{L}_G^h, G^h) = \prod_i \prod_{\upsilon \in \mathit{Val}(\Upsilon_{L_i})} \frac{\Gamma(\sum_{x_i} N'_{x_i|\upsilon})}{\Gamma(\sum_{x_i} N'_{x_i|\upsilon} + N(\upsilon))} \prod_{x_i} \frac{\Gamma(N'_{x_i|\upsilon} + N(x_i, \upsilon))}{\Gamma(N'_{x_i|\upsilon})}.$$

Once more we are faced with the problem of specifying a multitude of priors, that is, specifying $\Pr(\Theta_{X_i|\upsilon} \mid \mathcal{L}_G^h, G^h)$ for each possible combination of global and local structures. Our objective, as in the case tabular CPDs, is to set these priors from a prior distribution represented by a specific network B^P.

Recall that the values of the characteristic random variable are the partitions imposed by the local structure over $\mathit{Val}(\mathbf{Pa}_i)$. We make two assumptions regarding the priors and the groupings generated by this partition.

First, we assume that the prior for a value of the characteristic variable does not depend on the local structure. It depends only on the set of instances of the parents that are grouped by this particular value of the characteristic random variable. For example, consider two possible trees for the same CPD, one that tests first on Y and then on Z, and another that tests first on Z and then on Y. Our assumption requires that the leaves that correspond to $Y = y, Z = z$, be assigned the same prior in both trees.

Second, we assume that the vector of Dirichlet hyperparameters assigned to an element of the partition that corresponds to a union of several smaller partitions in another local structure is simply the sum of the vectors of Dirichlet hyperparameters assigned to these smaller partitions. Again, consider two trees, one that consists of a single leaf, and another that has one test at the root. This assumption requires that for each $x_i \in \mathit{Val}(X_i)$, the Dirichlet hyperparameter $N'_{x_i|\upsilon}$, where υ is the root in the first tree, is the sum of the $N'_{x_i|\upsilon'}$ for all the leaves in the second tree.

It is straightforward to show that if a prior distribution over structures, local structures, and parameters satisfies these assumptions and the assumptions of Heckerman et al. (1995a), then there must be a distribution P' and a positive real N' such that for any structure G and any choice of local structure $\mathcal{L}_G$ for G

$$\Pr(\Theta_{X_i|v} \mid \mathcal{L}_G^h, G^h) \sim \mathit{Dirichlet}(\{N' \cdot P'(x_i, \Upsilon_i^L = v) : x_i \in \mathit{Val}(X_i)\}) . \tag{11}$$

This result allows us to represent the prior information using a Bayesian network B' (that specifies the prior distribution P') and a positive real N'. From these two, we compute the Dirichlet hyperparameters for every hypothesis we need to evaluate during learning.

Finally, we note that we can use Schwarz's result (1978) to show that the MDL and BDe scores for local structure are asymptotically equivalent.

4.2. LEARNING PROCEDURES

Once we define the appropriate score, the learning task reduces to finding the network that maximizes the score, given the data. Unfortunately, this is an intractable problem. Chickering (1996) shows that finding the network (quantified with tabular CPDs) that maximizes the BDe score is NP-hard. Similar arguments also apply to learning with the MDL score. Moreover, there are indications that finding the optimal decision tree for a given family also is an NP-hard problem; see Quinlan and Rivest (1989). Thus, we suspect that finding a graph G and a set of local structures $\{L_1, \ldots, L_n\}$ that jointly maximize the MDL or BDe score is also an intractable problem.

A standard approach to dealing with hard optimization problems is heuristic search. Many search strategies can be applied. For clarity, we focus here on one of the simplest, namely *greedy hillclimbing*. In this strategy, we initialize the search with some network (e.g., the empty network) and repeatedly apply to the "current" candidate the local change (e.g., adding and removing edges) that leads to the largest improvement in the score. This "upward" step is repeated until a local maxima is reached, that is, no modification of the current candidate improves the score. Heckerman et al. (1995a) compare this greedy procedure with several more sophisticated search procedures. Their results indicate that greedy hillclimbing can be quite effective for learning Bayesian networks in practice.

The greedy hillclimbing procedure for learning network structure can be summarized as follows.

```
procedure LearnNetwork( G_0 )
    Let G_current ← G_0
    do
        Generate all successors S = {G_1, ..., G_n} of G_current
        ΔScore = max_{G∈S} Score(G) − Score(G_current)
        If ΔScore > 0 then
            Let G_current ← arg max_{G∈S} Score(G)
```

while($\Delta Score > 0$)
return $G_{current}$

The successors of the current structure are generated by adding an arc, removing an arc, and reversing the direction of an arc. (We consider only legal successors that do not involve a cycle.)

This greedy procedure is particularly efficient for learning Bayesian networks since the scores we use *decompose*. That is, both the MDL score and the (logarithm of the) BDe score have the the form $\sum_i Score(X_i \mid \mathbf{Pa}_i)$. Since the successors considered during the search modify at most two parent sets, we only need to recompute few terms to evaluate each successor. Moreover, we can cache these computations to get additional savings; see Bouckaert (1994) and Buntine (1991b).

When allowing local structured representations, we modify this loop by adding a learning operation before scoring each successor of G. This modification invokes a local search procedure that attempts to find (an approximation to) the best local structure for the each CPD. Since only one or two parent sets are modified in each successor, we invoke this procedure only for these CPDs.

The specific procedures used for learning default tables and decision trees are described next. Since these procedures are applied independently to each CPD, we fix the choice of X_i and of its parents $\mathbf{Pa}_i$ in the discussion below. Both procedures rely on additional decomposability properties of the score functions, in terms of the underlying partitions defined by the characteristic random variable. More precisely, the score of the data, given the local structure (i.e., , DL_{data} for MDL, and $\log \Pr(D \mid G^h, \mathcal{L}^h_G)$ for BDe), can be written as a sum

$$\sum_i \sum_{v \in Val(\Upsilon_{L_i})} Score(X_i \mid v),$$

where $Score(X_i \mid v)$ is a function of counts of the possible values X_i takes in these instances where $\Upsilon_{L_i} = v$. This decomposition implies that if we consider refining the local structure by replacing one partition (that corresponds to one value of Υ_{L_i}) by the union of several partitions, then we only need to reevaluate the terms that correspond to these new subpartitions.

We use a greedy strategy for inducing default tables. The procedure starts with a trivial default table containing only the default row. Then, it iteratively refines the default row, by finding the single row (i.e., assignment of values to the parents) that when represented explicitly leads to the biggest improvement in the score. This refinement can be done efficiently, since we need only to replace the term that corresponded to the previous default row with the sum of the terms that correspond to the new row and the new default row. This greedy expansion is repeated until no improvement in the score can be gained by adding another row. The procedure is summarized as follows.

procedure LearnDefault()
  Let $Rows(\mathcal{D}) \leftarrow \emptyset$
  do
    Let $r = \arg\max_{r \in Val(\mathbf{Pa}_i) - Rows(\mathcal{D})} Score(Rows(\mathcal{D}) \cup \{r\})$
    if $Score(Rows(\mathcal{D}) \cup \{r\}) < Score(Rows(\mathcal{D}))$ then
      return $Rows(\mathcal{D})$
    $Rows(\mathcal{D}) \leftarrow Rows(\mathcal{D}) \cup \{r\}$
end

For inducing decision trees, we adopt the approach outlined by Quinlan and Rivest (1989). The common wisdom in the decision-tree learning literature (e.g., Quinlan (1993)), is that greedy search of decision trees tends to become stuck at bad local minima. The approach of Quinlan and Rivest attempts to circumvent this problem using a two-phased approach. In the first phase we "grow" the tree in a top-down fashion. We start with the trivial tree consisting of one leaf, and add branches to it in a greedy fashion, until a maximal tree is learned. Note that in some stages of this growing phase, adding branches can lower the score: the rationale is that if we continue to grow these branches, we might improve the score. In the second phase, we remove harmful branches by "trimming" the tree in a bottom-up fashion. We now describe the two phases in more details.

In the first phase we grow a tree in a top-down fashion. We repeatedly replace a leaf with a subtree that has as its root some parent of X, say Y; and whose children are leaves, one for each value of Y. In order to decide on which parent Y we should *split* the tree, we compute the score of the tree associated with each parent, and select the parent that induces the best scoring tree. Since the scores we use are decomposable, we can compute the split in a *local* fashion by evaluating on the instances with respect to the training data that are compatible with the path from the root of the tree to the node that is being split. This recursive growing of the tree stops when the node has no training instances associated with it, the value of X is constant in the associated training set, or all the parents of X have been tested along the path leading to that node.

In the second phase, we trim the tree in a bottom-up manner. At each node we consider whether score of the subtree rooted at that node is better or equal to the score replacing that subtree by a leaf. If this is the case, then the subtree is trimmed and replaced with a leaf.

These two phases can be implemented by a simple recursive procedure, LearnTree, that receives a set of instances and returns the "best" tree for this set of instances.

```
procedure SimpleTree(Y)
    For y ∈ Val(Y), let l_y ← Λ (i.e., a leaf)
    return ⟨Y, {l_y : y ∈ Val(Y)}⟩
end
procedure LearnTree(D)
    if D = ∅ or X_i is homogeneous in D then
        return Λ.
    // Growing phase
    Let Y_split = arg max_{Y ∈ Pa_i} Score(SimpleTree(Y) | D)
    for y ∈ Val(Y_split)
        Let D_y = {u_i ∈ D : Y_split = y in u_i}
        Let T_y = ExpandTree( Λ, D_y)
    let T = ⟨Y_split, {T_y : y ∈ Val(Y)}⟩
    // Trimming phase
    if Score(Λ | D) > Score(T | D) then
        return Λ
    else
        return T
end
```

5. Experimental Results

The main purpose of our experiments is to confirm and quantify the hypothesis stated in the introduction: A learning procedure that learns local structures for the CPDs will induce more accurate models for two reasons: 1) fewer parameters will lead to a more reliable estimation, and 2) flexible penalty for larger families will result in network structures that are better approximations to the real (in)dependencies in the underlying distribution.

The experiments compared networks induced with table-based, tree-based, and default-based procedures, where an X-based procedure learns networks with X as the representation of CPDs. We ran experiments using both the MDL score and the BDe score. When using the BDe score, we also needed to provide a prior distribution and equivalent sample size. In all of our experiments, we used a uniform prior distribution, and examined several settings of the equivalent sample size N'. All learning procedures were based on the same search method discussed in Section 4.2.

We ran experiments with several variations, including different settings of the BDe prior equivalent size and different initialization points for the search procedures. These experiments involved learning approximately 15,000 networks. The results are summarized below.

TABLE 1. Description of the networks used in the experiments.

Name	Description	n	$\|\|\mathbf{U}\|\|$	$\|\Theta\|$
Alarm	A network by medical experts for monitoring patients in intensive care (Beinlich et al., 1989).	37	$2^{53.95}$	509
Hailfinder	A network for modeling summer hail in northeastern Colorado (`http://www.lis.pitt.edu/~dsl/hailfinder`).	56	$2^{106.56}$	2656
Insurance	A network for classifying insurance applications (Russell et al., 1995).	27	$2^{44.57}$	1008

5.1. METHODOLOGY

The data sets used in the experiments were sampled from three Bayesian networks whose main characteristics are described in Table 1. From each of these networks we sampled training sets of sizes—250, 500, 1000, 2000, 4000, 8000, 16000, 24000, and 32000 instances—and ran the learning procedures on them. The learning procedures received only the data sets, and did not have access to the generating network. In order to increase the accuracy of the results, we repeated the experiment with 10 (independently sampled) sets of training data. In all of the experiments, the methods we compared received as input the same training data.

By virtue of having a golden model in each experiment, represented by the original networks, we could precisely quantify the error between the induced models and the original model. We were also able to quantify the effect of the local structures on the parameter estimation and the structure selection.

5.2. MEASURES OF ERROR

As the main measurement of error we use the *entropy distance* (also known as *Kullbak-Leibler divergence* and *relative entropy*) from the generating distribution to the induced distribution. The entropy distance from a distribution P to an approximation Q is defined as

$$D(P\|Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)}.$$

This quantity is a measure of the inefficiency incurred by assuming that the distribution is Q when the real distribution is P. Note that the entropy distance is not symmetric, i.e., $D(P\|Q)$ is not equal in general to $D(Q\|P)$. Another important property of the entropy distance function is that $D(P\|Q) \geq 0$, where equality holds if and only if $P = Q$.

There are several possible justifications for using the entropy distance. On the axiomatic side, Shore and Johnson (1980) suggest several desirable properties of approximation measures, and show that entropy distance is the only function that satisfies all of them. There are also motivating examples from data compression and gambling. In both examples, the entropy distance measures the *loss* incurred by using the distribution Q instead of the true distribution P (e.g., additional bits needed or expected monetary losses). We refer the reader to Cover and Thomas (1991) for a discussion and a detailed analysis of these examples.

Measuring the entropy distance of the induced networks allows us to compare the generalization error of the different procedures. We are also interested in assessing the separate influences of the parameter estimation and the induced network structure on this error.

Let G be a network structure. We define the *inherent error* of G with respect to a target distribution P^* as

$$D_{\text{struct}}(P^* \| G) = \min_{\mathcal{L}} D(P^* \| (G, \mathcal{L})).$$

The inherent error of G is the smallest error achievable by any possible choice of CPDs $\mathcal{L}$ for G. Thus, even if we can find the "best" possible parameters for G, we still cannot hope to get a smaller error than $D_{\text{struct}}(P^* \| G)$.

As it turns out, this measure of error can be evaluated by means of a closed-form equation. As we might expect, the best CPDs for G are those where the conditional distribution of X_i, given $\mathbf{Pa}_i$, is identical to $P^*(X_i \mid \mathbf{Pa}_i)$.

Proposition 5.1: *Let G be a network structure and let P^* be a distribution. Then $D_{\text{struct}}(P^* \| G) = D(P^* \| (G, \mathcal{L}^*))$, where $\mathcal{L}^*$ is such that $P(X_i \mid \mathbf{Pa}_i) = P^*(X_i \mid \mathbf{Pa}_i)$ for all i.*

An alternative way of thinking about the inherent error of a network structure G, is as a measure of how "reasonable" are the independence assumptions encoded in G. We can attempt to measure the error of the network structure by estimating to what degree each of these independencies is violated in P^*. One way of measuring the strength of the dependency between variables is the measure of *conditional mutual information.* Let $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ be three sets of variables; the conditional mutual information between $\mathbf{X}$ and $\mathbf{Y}$, given $\mathbf{Z}$, is defined as

$$I_P(\mathbf{X}; \mathbf{Y} \mid \mathbf{Z}) = H_P(\mathbf{X} \mid \mathbf{Z}) - H_P(\mathbf{X} \mid \mathbf{Y}, \mathbf{Z}).$$

Intuitively, this term measures how much the knowledge of $\mathbf{Y}$ helps us compress $\mathbf{X}$ when we already know $\mathbf{Z}$. It well known that $I_P(\mathbf{X}; \mathbf{Y} \mid \mathbf{Z}) \geq 0$, and that $I_P(\mathbf{X}; \mathbf{Y} \mid \mathbf{Z}) = 0$, if and only if $\mathbf{X}$ is independent of $\mathbf{Y}$, given $\mathbf{Z}$ (Cover and Thomas, 1991).

Using the mutual information as a quantitative measure of strength of dependencies, we can measure the extent to which the independence assumptions represented in G are violated in the real distribution. This suggests that we evaluate this measure for all conditional independencies represented by G. However, many of these independence assumptions "overlap" in the sense that they imply each other. Thus, we need to find a minimal set of independencies that imply all the other independencies represented by G. Pearl (1988) shows how to construct such a minimal sets of independence. Assume that the variable ordering $X_1, \ldots, X_n$ is consistent with the arc direction in G (i.e., if X_i is a parent of X_j, then $i < j$). If, for every i, X_i is independent of $\{X_1, \ldots X_{i-1}\} - \mathbf{Pa}_i$, given $\mathbf{Pa}_i$, then using the chain rule we find that P can be factored as in Equation 1. As a consequence, we find that this set of independence assumptions implies all the independence assumptions that are represented by G. Starting with different consistent orderings, we get different minimal sets of assumptions. However, the next proposition shows that evaluating the error of the model with respect to any of these sets leads to the same answer.

Proposition 5.2: *Let G be a network structure, $X_1, \ldots, X_n$ be a variable ordering consistent with arc direction in G, and P^* be a distribution. Then*

$$D_{\text{struct}}(P^* \| G) = \sum_i I_{P^*}(X_i; \{X_1, \ldots X_{i-1}\} - \mathbf{Pa}_i \mid \mathbf{Pa}_i) \ .$$

This proposition shows that $D_{\text{struct}}(P^* \| G) = 0$ if and only if G is an *I-map* of P^*; that is, all the independence statements encoded in G are also true of P^*. Small values of $D_{\text{struct}}(P^* \| G)$ indicate that while G is not an I-map of P^*, the dependencies not captured by G are "weak." We note that $D_{\text{struct}}(P^* \| G)$ is a one-sided error measure, in the sense that it penalizes structures for representing wrong independence statements, but does not penalize structures for representing redundant dependence statements. In particular, *complete* network structures (i.e., ones to which we cannot add edges without introducing cycles) have no inherent error, since they do not represent any conditional independencies.

We can postulate now that the difference between the overall error (as measured by the entropy distance) and the inherent error is due to errors introduced in the estimation of the CPDs. Note that when we learn a local structure, some of this additional error may be due to the induction of an inappropriate local structure, such as a local structure that makes assumptions of context-specific independencies that do not hold in the target distribution. As with global structure, we can measure the inherent error in the local structure learned. Let G be a network structure, and let $S_{L_1}, \ldots, S_{L_n}$ be structures for the CPDs of G. The *inherent local error* of

G and $S_{L_1}, \ldots, S_{L_n}$ is

$$D_{\text{local}}(P^* \| G, \{S_{L_1}, \ldots, S_{L_n}\}) = \min_{\Theta_{L_1}, \ldots, \Theta_{L_n}} D(P \| (G, \{(S_{L_1}, \Theta_{L_1}), \ldots, (S_{L_n}, \Theta_{L_n})\})).$$

From the above, we get the following expected generalization of Proposition 5.1.

Proposition 5.3: *Let G be a network structure, let $S_{L_1}, \ldots, S_{L_n}$ be local structure for the CPDs of G, and let P^* be a distribution. Then*

$$D_{\text{local}}(P^* \| G, \{S_{L_1}, \ldots, S_{L_n}\}) = D(P^* \| (G, \mathcal{L}^*)),$$

where $\mathcal{L}^*$ is such that $P(X_i \mid \Upsilon_{L_i}) = P^*(X_i \mid \Upsilon_{L_i})$ for all i.

From the definitions of inherent error above it follows that for any network $B = (G, \mathcal{L})$,

$$D(P^* \| P_B) \geq D_{\text{local}}(P^* \| G, \{S_{L_1}, \ldots, S_{L_n}\}) \geq D_{\text{struct}}(P^* \| G).$$

Using these measures in the evaluation of our experiments, we can measure the "quality" of the global independence assumptions made by a network structure (D_{struct}), the quality of the local and global independence assumptions made by a network structure and a local structure (D_{local}), and the total error, which also includes the quality of the parameters.

5.3. RESULTS

We want to characterize the error in the induced models as a function of the number of samples used by the different learning algorithms for the induction. Thus, we plot *learning curves* where the x-axis displays the number of training instances N, and the y-axis displays the error of the learned model. In general, these curves exhibit exponential decrease in the error. This makes visual comparisons between different learning procedures hard, since the differences in the large-sample range ($N \geq 8000$) are obscured, and, when a logarithmic scale is used for the x-axis, the differences at the small-sample range are hard to visualize. See for example Figure 3(a) and (b).

To address this problem, we propose a normalization of these curves, motivated by the theoretical results of Friedman and Yakhini (1996). They show that learning curves for Bayesian networks generally behave as a linear function of $\frac{\log N}{N}$. Thus, we plot the error scaled by $\frac{N}{\log N}$. Figure 5(a) shows the result of the application of this normalization to the curves in Figure 3. Observe that the resulting curves are roughly constant. The thin dotted diagonal lines in Figure 5(a) correspond to the lines of constant error in Figure 3. We plot these lines for entropy distances of $1/2^i$ for $i = 0, \ldots, 6$.

(a)

(b)

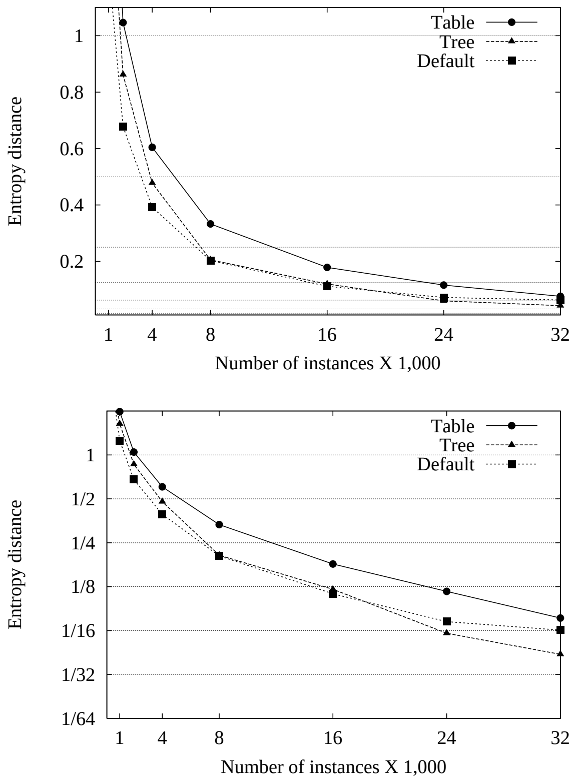


Figure 3. (a) Error curves showing the entropy distance achieved by procedures using the MDL score in the Alarm domain. The x-axis displays the number of training instances N, and the y-axis displays the entropy distances of the induced network. (b) The same curves with logarithmic y-axis. Each point is the average of learning from 10 independent data sets.

Figures 4 and 5 display the entropy distance of networks learned via the BDe and MDL scores (Table 2 summarizes these values.) In all the experiments, the learning curves appear to converge to the target distribution: eventually they would intersect the dotted line of ϵ entropy distance for all $\epsilon > 0$. Moreover, all of them appear to (roughly) conform to the behavior specified by the results of Friedman and Yakhini.

With respect to the entropy distance, tree-based procedures performed better in all our experiments than table-based procedures. With few exceptions, the default-based procedures also performed better than the table-based procedures in the Alarm and Insurance domains. The default-based methods performed poorly in the Hailfinder domain.

As a general rule, we see a constant gap in the the curves corresponding

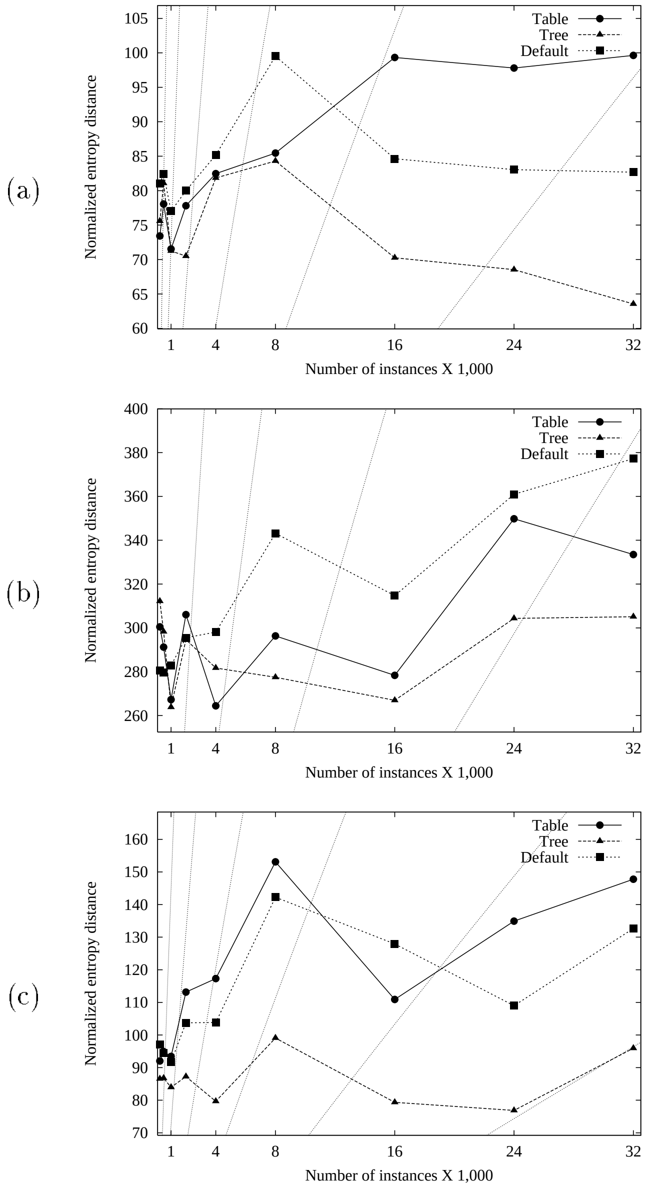


Figure 4. Normalized error curves showing the entropy distance achieved by procedures using the BDe (with $N' = 1$) score in the (a) Alarm domain, (b) Hailfinder domain, and (c) Insurance domain. The x-axis displays the number of training instances N, and the y-axis displays the normalized entropy distances of the induced network (see Section 5.2).

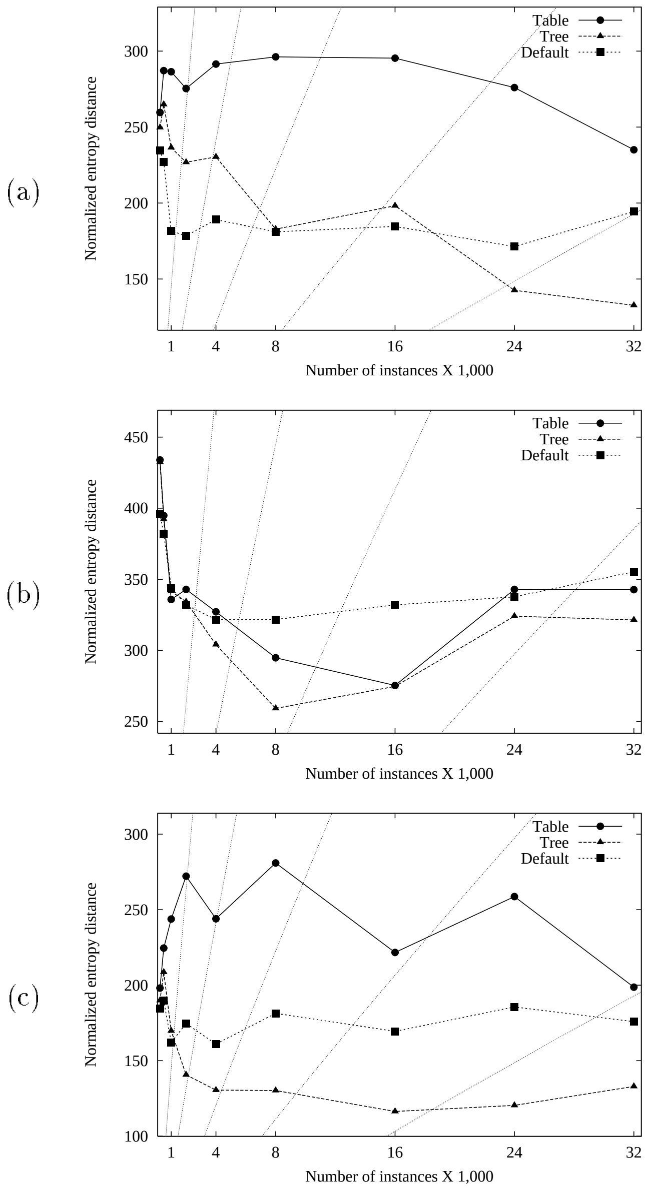


Figure 5. Normalized error curves showing the entropy distance achieved by procedures using the MDL score in the (a) Alarm domain, (b) Hailfinder domain, and (c) Insurance domain.

to different representations. Thus, for a fixed N, the error of the procedure representing local structure is a constant fraction of the error of the corresponding procedure that does not represent local structure (i.e., learns tabular CPDs). For example, in Figure 4(a) we see that in the large-sample region (e.g., $N \geq 8000$), the errors of procedures that use trees and default tables are approximately 70% and 85% (respectively) of the error of the table-based procedures. In Figure 5(c) the corresponding ratios are 50% and 70%.

Another way of interpreting these results is obtained by looking at the number of instances needed to reach a particular error rate. For example, In Figure 4(a), the tree-based procedure reaches the error level of $\frac{1}{32}$ with approximately 23,000 instances. On the other hand, the table-based procedure barely reaches that error level with 32,000 instances. Thus, if we want to ensure this level of performance, we would need to supply the table-based procedure with 9,000 additional instances. This number of instances might be unavailable in practice.

We continued our investigation by examining the network structures learned by the different procedures. We evaluated the inherent error, D_{struct}, of the structures learned by the different procedures. In all of our experiments, the inherent error of the network structures learned via tree-based and default-based procedures is smaller than the inherent error of the networks learned by the corresponding table-based procedure. For example, examine the D_{struct} column in Tables 3 and 4. From these results, we conclude that the network structures learned by procedures using local representations make fewer mistaken assumptions of global independence, as predicted by our main hypothesis.

Our hypothesis also predicts that procedures that learn local representation are able to assess fewer parameters by making local assumptions of independence in the CPDs. To illustrate this, we measured the inherent local error, D_{local}, and the number of parameters needed to quantify these networks. As we can see in Tables 3 and 4, the networks learned by these procedures exhibit smaller inherent error, D_{struct}; but they require fewer parameters, and their inherent local error, D_{local}, is roughly the same as that of networks learned by the table-based procedures. Hence, instead of making global assumptions of independence, the local representation procedures make the local assumptions of independence that better capture the regularities in the target distribution and require fewer parameters. As a consequence, the parameter estimation for these procedures is more accurate.

Finally, we investigated how our conclusions depend on the particular choices we made in the experiments. As we will see, the use of local structure leads to improvements regardless of these choices. We examined two aspects of the learning process: the choice of the parameters for the priors and in the search procedure.

We start by looking at the effect of changing the equivalent sample size N'. Heckerman et al. (1995a) show that the choice of N' can have drastic effects on the quality of the learned networks. On the basis of on their experiments in the Alarm domain, Heckerman et al. report that $N' = 5$ achieves the best results. Table 5 shows the effect of changing N' from 1 to 5 in our experiments. We see that the choice of N' influences the magnitude of the errors in the learned networks, and the sizes of the error gaps between the different methods. Yet these influences do not suggest any changes on the benefits of local structures.

Unlike the BDe score, the MDL score does not involve an explicit choice of priors. Nonetheless, we can use Bayesian averaging to select the parameters for the structures that have been learned by the MDL score, as opposed to using maximum likelihood estimates. In Table 6 we compare the error between the maximum likelihood estimates and Bayesian averaging with $N' = 1$. As expected, averaging leads to smaller errors in the parameter estimation, especially for small sample sizes. However, with the exception of the Alarm domain, Bayesian averaging does not improve the score for large samples (e.g., $N = 32,000$). We conclude that even though changing the parameter estimation technique may improve the score in some instances, it does not change our basic conclusions.

Finally, another aspect of the learning process that needs further investigation is the heuristic search procedure. A better search technique can lead to better induced models as illustrated in the experiments of Heckerman et al. (1995a). In our experiments we modified the search by initializing the greedy search procedure with a more informed starting point. Following Heckerman et al. (1995a) we used the *maximal branching* as a starting state for the search. A maximal branching network is one of the highest-scoring network among these where $|\mathbf{Pa}_i| \leq 1$ for all i. A maximal branching can be found in an efficient manner (e.g., in low-order polynomial time) (Heckerman et al., 1995a). Table 7 reports the results of this experiment. In the Alarm domain, the use of maximal branching as an initial point led to improvements in all the learning procedures. On the other hand, in the Insurance domain, this choice of for a starting point led to a worse error. Still, we observe that the conclusions described above regarding the use of local structure held for these runs as well.

6. Conclusion

The main contribution of this article is the introduction of structured representations of the CPDs in the learning process, the identification of the benefits of using these representations, and the empirical validation of our hypothesis. As we mentioned in the introduction (Section 1), we are not the first to consider efficient representations for the CPDs in the context of learning. However, to the best of our knowledge, we are the first to consider and demonstrate the effects that these representations may have on the learning of the global structure of the network.

In this paper we have focused on the investigation of two fairly simple, structured representations of CPDs: trees and default tables. There are certainly many other possible representation of CPDs, based, for example, on decision graphs, rules, and CNF formulas: see Boutilier et al. (1996). Our choice was mainly due to the availability of efficient computational tools for learning the representations we use. The refinement of the methods studied in this paper to incorporate these representations deserves further attention. In the machine learning literature, there are various approaches to learning trees, all of which can easily be incorporated in the learning procedures for Bayesian networks. In addition, certain interactions among the search procedures for global and local structures can be exploited, to reduce the computational cost of the learning process. We leave these issues for future research.

It is important to distinguish between the local representations we examine in this paper and the noisy-or and logistic regression models that have been examined in the literature. Both noisy-or and logistic regression (as applied in the Bayesian network literature) attempt to estimate the CPD with a *fixed* number of parameters. This number is usually linear in the number of parents in the CPD. In cases where the target distribution does not satisfy the assumptions embodied by these models, the estimates of CPDs produced by these methods can arbitrarily diverge from the target distribution. On the other hand, our local representations involve learning the structure of the CPD, which can range from a lean structure with few parameters to a complex structure with an exponential number of parameters. Thus, our representations can scale up to accommodate the complexity of the training data. This ensures that, in theory, they are asymptotically correct: given enough samples, they will construct a close approximation of the target distribution.

In conclusion, we have shown that the induction of local structured representation for CPDs significantly improves the performance of procedures for learning Bayesian networks. In essence, this improvement is due to the fact that we have changed the *bias* of the learning procedure to reflect the nature of the distribution in the data more accurately.

Acknowledgments

The authors are grateful to an anonymous reviewer and to Wray Buntine and David Heckerman for their comments on previous versions of this paper and for useful discussions relating to this work.

Part of this research was done while both authors were at the Rockwell Science Center, [4] Palo Alto Laboratory. Nir Friedman was also at Stanford University at the time. The support provided by Rockwell and Stanford University is gratefully acknowledged. In addition, Nir Friedman was supported in part by an IBM graduate fellowship and NSF Grant IRI-95-03109.

A preliminary version of this article appeared in the *Proceedings,* 12^{th} *Conference on Uncertainty in Artificial Intelligence*, 1996.

References

I. Beinlich, G. Suermondt, R. Chavez, and G. Cooper. The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In *Proc. 2'nd European Conf. on AI and Medicine.* Springer-Verlag, Berlin, 1989.

R. R. Bouckaert. Properties of Bayesian network learning algorithms. In R. López de Mantarás and D. Poole, editors, *Proc. Tenth Conference on Uncertainty in Artificial Intelligence (UAI '94)*, pages 102–109. Morgan Kaufmann, San Francisco, CA, 1994.

C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in Bayesian networks. In E. Horvitz and F. Jensen, editors, *Proc. Twelfth Conference on Uncertainty in Artificial Intelligence (UAI '96)*, pages 115–123. Morgan Kaufmann, San Francisco, CA, 1996.

W. Buntine. *A theory of learning classification rules.* PhD thesis, University of Technology, Sydney, Australia, 1991.

W. Buntine. Theory refinement on Bayesian networks. In B. D. D'Ambrosio, P. Smets, and P. P. Bonissone, editors, *Proc. Seventh Annual Conference on Uncertainty Artificial Intelligence (UAI '92)*, pages 52–60. Morgan Kaufmann, San Francisco, CA, 1991.

W. Buntine. Learning classification trees. In D. J. Hand, editor, *Artificial Intelligence Frontiers in Statistics*, number III in AI and Statistics. Chapman & Hall, London, 1993.

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.

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

T. M. Cover and J. A. Thomas. *Elements of Information Theory.* John Wiley & Sons, New York, 1991.

M. H. DeGroot. *Optimal Statistical Decisions.* McGraw-Hill, New York, 1970.

F. J. Diez. Parameter adjustment in Bayes networks: The generalized noisy or-gate. In D. Heckerman and A. Mamdani, editors, *Proc. Ninth Conference on Uncertainty in Artificial Intelligence (UAI '93)*, pages 99–105. Morgan Kaufmann, San Francisco, CA, 1993.

N. Friedman and Z. Yakhini. On the sample complexity of learning Bayesian networks. In E. Horvitz and F. Jensen, editors, *Proc. Twelfth Conference on Uncertainty in*

[4] All products and company names mentioned in this article are the trademarks of their respective holders.

Artificial Intelligence (UAI '96). Morgan Kaufmann, San Francisco, CA, 1996.

D. Heckerman and J. S. Breese. A new look at causal independence. In R. López de Mantarás and D. Poole, editors, *Proc. Tenth Conference on Uncertainty in Artificial Intelligence (UAI '94)*, pages 286–292. Morgan Kaufmann, San Francisco, CA, 1994.

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

D. Heckerman. A tutorial on learning Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, 1995.

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

R. Musick. *Belief Network Induction*. PhD thesis, University of California, Berkeley, CA, 1994.

R. M. Neal. Connectionist learning of belief networks. *Artificial Intelligence*, 56:71–113, 1992.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Francisco, CA, 1988.

J. R. Quinlan and R. Rivest. Inferring decision trees using the minimum description length principle. *Information and Computation*, 80:227–248, 1989.

J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Francisco, CA, 1993.

J. Rissanen. *Stochastic Complexity in Statistical Inquiry*. World Scientific, River Edge, NJ, 1989.

S. Russell, J. Binder, D. Koller, and K. Kanazawa. Local learning in probabilistic networks with hidden variables. In *Proc. Fourteenth International Joint Conference on Artificial Intelligence (IJCAI '95)*, pages 1146–1152. Morgan Kaufmann, San Francisco, CA, 1995.

G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

J. E. Shore and R. W. Johnson. Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy. *IEEE Transactions on Information Theory*, IT-26(1):26–37, 1980.

D. J. Spiegelhalter and S. L. Lauritzen. Sequential updating of conditional probabilities on directed graphical structures. *Networks*, 20:579–605, 1990.

S. Srinivas. A generalization of the noisy-or model. In D. Heckerman and A. Mamdani, editors, *Proc. Ninth Conference on Uncertainty in Artificial Intelligence (UAI '93)*, pages 208–215. Morgan Kaufmann, San Francisco, CA, 1993.

C. Wallace and J. Patrick. Coding decision trees. *Machine Learning*, 11:7–22, 1993.

TABLE 2. Summary of entropy distance for networks learned by the procedure using the MDL score and BDe score with $N' = 1$.

Domain	Size (× 1,000)	MDL Score Table	MDL Score Tree	MDL Score Defualt	BDe Score Table	BDe Score Tree	BDe Score Default
Alarm	0.25	5.7347	5.5148	5.1832	1.6215	1.6692	1.7898
	0.50	3.5690	3.2925	2.8215	0.9701	1.0077	1.0244
	1.00	1.9787	1.6333	1.2542	0.4941	0.4922	0.5320
	2.00	1.0466	0.8621	0.6782	0.2957	0.2679	0.3040
	4.00	0.6044	0.4777	0.3921	0.1710	0.1697	0.1766
	8.00	0.3328	0.2054	0.2034	0.0960	0.0947	0.1118
	16.00	0.1787	0.1199	0.1117	0.0601	0.0425	0.0512
	24.00	0.1160	0.0599	0.0720	0.0411	0.0288	0.0349
	32.00	0.0762	0.0430	0.0630	0.0323	0.0206	0.0268
Hailfinder	0.25	9.5852	9.5513	8.7451	6.6357	6.8950	6.1947
	0.50	4.9078	4.8749	4.7475	3.6197	3.7072	3.4746
	1.00	2.3200	2.3599	2.3754	1.8462	1.8222	1.9538
	2.00	1.3032	1.2702	1.2617	1.1631	1.1198	1.1230
	4.00	0.6784	0.6306	0.6671	0.5483	0.5841	0.6181
	8.00	0.3312	0.2912	0.3614	0.3329	0.3117	0.3855
	16.00	0.1666	0.1662	0.2009	0.1684	0.1615	0.1904
	24.00	0.1441	0.1362	0.1419	0.1470	0.1279	0.1517
	32.00	0.1111	0.1042	0.1152	0.1081	0.0989	0.1223
Insurance	0.25	4.3750	4.1940	4.0745	2.0324	1.9117	2.1436
	0.50	2.7909	2.5933	2.3581	1.1798	1.0784	1.1734
	1.00	1.6841	1.1725	1.1196	0.6453	0.5799	0.6335
	2.00	1.0343	0.5344	0.6635	0.4300	0.3316	0.3942
	4.00	0.5058	0.2706	0.3339	0.2432	0.1652	0.2153
	8.00	0.3156	0.1463	0.2037	0.1720	0.1113	0.1598
	16.00	0.1341	0.0704	0.1025	0.0671	0.0480	0.0774
	24.00	0.1087	0.0506	0.0780	0.0567	0.0323	0.0458
	32.00	0.0644	0.0431	0.0570	0.0479	0.0311	0.0430

TABLE 3. Summary of inherent error, inherent local error, and number of parameters for the networks learned by the table-based and the tree-based procedures using the BDe score with $N' = 1$.

Domain	Size (× 1,000)	Table D	Table $D_{\text{struct}}/D_{\text{local}}$	Table Param	Tree D	Tree D_{local}	Tree D_{struct}	Tree Param
Alarm	1	0.4941	0.1319	570	0.4922	0.1736	0.0862	383
	4	0.1710	0.0404	653	0.1697	0.0570	0.0282	453
	16	0.0601	0.0237	702	0.0425	0.0154	0.0049	496
	32	0.0323	0.0095	1026	0.0206	0.0070	0.0024	497
Hailfinder	1	1.8462	1.2166	2066	1.8222	1.1851	1.0429	1032
	4	0.5483	0.3434	2350	0.5841	0.3937	0.2632	1309
	16	0.1684	0.1121	2785	0.1615	0.1081	0.0758	1599
	32	0.1081	0.0770	2904	0.0989	0.0701	0.0404	1715
Insurance	1	0.6453	0.3977	487	0.5799	0.3501	0.2752	375
	4	0.2432	0.1498	724	0.1652	0.0961	0.0654	461
	16	0.0671	0.0377	938	0.0480	0.0287	0.0146	525
	32	0.0479	0.0323	968	0.0311	0.0200	0.0085	576

TABLE 4. Summary of inherent error, inherent local error, and number of parameters for the networks learned by the table-based and tree-based procedures using the MDL score.

Domain	Size (× 1,000)	Table D	Table $D_{\text{struct}}/D_{\text{local}}$	Table Param	Tree D	Tree D_{local}	Tree D_{struct}	Tree Param
Alarm	1	1.9787	0.5923	361	1.6333	0.4766	0.3260	289
	4	0.6044	0.2188	457	0.4777	0.1436	0.0574	382
	16	0.1787	0.0767	639	0.1199	0.0471	0.0189	457
	32	0.0762	0.0248	722	0.0430	0.0135	0.0053	461
Hailfinder	1	2.3200	1.0647	1092	2.3599	1.1343	0.9356	1045
	4	0.6784	0.4026	1363	0.6306	0.3663	0.2165	1322
	16	0.1666	0.1043	1718	0.1662	0.1107	0.0621	1583
	32	0.1111	0.0743	1864	0.1042	0.0722	0.0446	1739
Insurance	1	1.6841	1.0798	335	1.1725	0.5642	0.4219	329
	4	0.5058	0.3360	518	0.2706	0.1169	0.0740	425
	16	0.1341	0.0794	723	0.0704	0.0353	0.0187	497
	32	0.0644	0.0355	833	0.0431	0.0266	0.0140	544

TABLE 5. Summary of entropy distance for procedures that use the BDe score with $N' = 1$ and $N' = 5$.

Domain	Size (× 1,000)	$N' = 1$ Table	$N' = 1$ Tree	$N' = 1$ Default	$N' = 5$ Table	$N' = 5$ Tree	$N' = 5$ Default
Alarm	1	0.4941	0.4922	0.5320	0.3721	0.3501	0.3463
	4	0.1710	0.1697	0.1766	0.1433	0.1187	0.1308
	16	0.0601	0.0425	0.0512	0.0414	0.0352	0.0435
	32	0.0323	0.0206	0.0268	0.0254	0.0175	0.0238
Hailfinder	1	1.8462	1.8222	1.9538	1.4981	1.5518	1.6004
	4	0.5483	0.5841	0.6181	0.4574	0.4859	0.5255
	16	0.1684	0.1615	0.1904	0.1536	0.1530	0.1601
	32	0.1081	0.0989	0.1223	0.0996	0.0891	0.0999
Insurance	1	0.6453	0.5799	0.6335	0.5568	0.5187	0.5447
	4	0.2432	0.1652	0.2153	0.1793	0.1323	0.1921
	16	0.0671	0.0480	0.0774	0.0734	0.0515	0.0629
	32	0.0479	0.0311	0.0430	0.0365	0.0284	0.0398

TABLE 6. Summary of entropy distance for procedures that use the MDL score for learning the structure and local structure combined with two methods for parameter estimation.

Domain	Size (× 1,000)	Maximum Likelihood Table	Maximum Likelihood Tree	Maximum Likelihood Default	Bayesian, $N' = 1$ Table	Bayesian, $N' = 1$ Tree	Bayesian, $N' = 1$ Default
Alarm	1	1.9787	1.6333	1.2542	0.8848	0.7495	0.6015
	4	0.6044	0.4777	0.3921	0.3251	0.2319	0.2229
	16	0.1787	0.1199	0.1117	0.1027	0.0730	0.0779
	32	0.0762	0.0430	0.0630	0.0458	0.0267	0.0475
Hailfinder	1	2.3200	2.3599	2.3754	1.7261	1.7683	1.8047
	4	0.6784	0.6306	0.6671	0.5982	0.5528	0.6091
	16	0.1666	0.1662	0.2009	0.1668	0.1586	0.1861
	32	0.1111	0.1042	0.1152	0.1133	0.0964	0.1120
Insurance	1	1.6841	1.1725	1.1196	1.1862	0.7539	0.8082
	4	0.5058	0.2706	0.3339	0.3757	0.1910	0.2560
	16	0.1341	0.0704	0.1025	0.1116	0.0539	0.0814
	32	0.0644	0.0431	0.0570	0.0548	0.0368	0.0572

TABLE 7. Summary of entropy distance for two methods for initializing the search, using the the BDe score with $N' = 1$.

		Empty Network			Maximal Branching Network		
Domain	Size ($\times$ 1,000)	Table	Tree	Default	Table	Tree	Default
Alarm	1	0.4941	0.4922	0.5320	0.4804	0.5170	0.4674
	4	0.1710	0.1697	0.1766	0.1453	0.1546	0.1454
	16	0.0601	0.0425	0.0512	0.0341	0.0350	0.0307
	32	0.0323	0.0206	0.0268	0.0235	0.0191	0.0183
Hailfinder	1	1.8462	1.8222	1.9538	1.7995	1.7914	1.9972
	4	0.5483	0.5841	0.6181	0.6220	0.6173	0.6633
	16	0.1684	0.1615	0.1904	0.1782	0.1883	0.1953
	32	0.1081	0.0989	0.1223	0.1102	0.1047	0.1162
Insurance	1	0.6453	0.5799	0.6335	0.6428	0.6350	0.6502
	4	0.2432	0.1652	0.2153	0.2586	0.2379	0.2242
	16	0.0671	0.0480	0.0774	0.1305	0.0914	0.1112
	32	0.0479	0.0311	0.0430	0.0979	0.0538	0.0856