Learning with Knowledge from Multiple Experts

Matthew Richardson MATTR@CS.WASHINGTON.EDU
Pedro Domingos PEDROD@CS.WASHINGTON.EDU
Department of Computer Science and Engineeering, University of Washington, Seattle, WA 98195-2350, USA

Abstract

The use of domain knowledge in a learner can greatly improve the models it produces. However, high-quality expert knowledge is very difficult to obtain. Traditionally, researchers have assumed that knowledge comes from a single self-consistent source. A little-explored but often more feasible alternative is to use multiple weaker sources. In this paper we take a step in this direction by developing a method for learning the structure of a Bayesian network from multiple experts. Data is then used to refine the structure and estimate parameters. A simple analysis shows that even relatively few noisy experts can produce high-quality knowledge when combined. Experiments with real and simulated experts in a variety of domains show the benefits of this approach.

1. Introduction

The accuracy of a learner can be greatly improved by explicitly incorporating domain knowledge into it. A learner's output will also generally be much more comprehensible if the learner takes existing knowledge into account (Pazzani et al., 1997). Recognizing this, extensive research on combining empirical learning with domain knowledge has been carried out (e.g., Bergadano and Giordana (1988); Pazzani and Kibler (1992); Ourston and Mooney (1994); Towell and Shavlik (1994)). However, the use of knowledge-intensive learning has not become widespread in practice. The chief cause of this is the high cost and difficulty of obtaining domain knowledge (Scott et al., 1991; Marcus, 1989; Henrion, 1987). Typically, the knowledge provided by an expert is initially very buggy and incomplete, and a laborious process of interaction with the knowledge engineer is required to refine it to the point where it becomes useful. As a result, most learners in wide use today employ only very weak forms of knowledge (e.g., a similarity or smoothness bias), and this can significantly limit their accuracy, particularly in large, complex domains.

In this paper we attempt to find a way out of this dilemma by exploring a new approach to knowledge-intensive learning, inspired by the success of model ensemble methods like bagging (Breiman, 1996), boosting (Freund & Schapire, 1996) and stacking (Wolpert, 1992). In essence, these methods combine many "weak" (i.e., not very accurate) models into a "strong" one. This can produce surprisingly large improvements in accuracy. Our approach is to combine the knowledge of many "weak" experts into a "strong" collective knowledge base. In particular, in this paper we propose a method for combining statements about the structure of a Bayesian network from multiple experts into a single structure that more closely captures the dependencies in the domain. This structure is then refined, and its parameters estimated, using standard Bayesian network learning algorithms. The core of our method is to postulate a simple generative model for what an expert says given the actual dependencies in the domain, and use Bayes' theorem to obtain a posterior distribution over those dependencies given what the experts say.

The necessary knowledge can be gathered from experts using traditional interview methods. A more recent alternative is to solicit it over an organization's intranet, or from the relevant community of interest over the Internet, with an appropriate interface for knowledge entry. This avenue makes it possible to gather knowledge from more contributors at lower cost. The knowledge obtained in this way is likely to be more diverse, but also of more variable quality, than that gathered by traditional means. This makes it well suited to our approach. Its promise is shown by the success of online forecasting markets (Pennock et al., 2001), knowledge-sharing sites like epinions.com (Frauenfelder, 2000), and PC manufacturers' Web sites for mutual customer help (Morris, August 2001). The research described in this paper is part of our CKB (Collective Knowledge Bases) project, which is developing tools for the construction

Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003.

of large knowledge bases by collaboration over the Internet, and for their refinement using machine learning (Richardson & Domingos, 2003). Related efforts include the Open Mind Initiative (www.openmind.org (Stork, 2000)) and OpenCyc (www.opencyc.org), the open-source version of Cyc (Lenat & Guha, 1990).

The combination of beliefs from multiple experts has received some attention in the statistical literature (Lindley, 1985; French, 1985; Genest & Zidek, 1986; Pennock & Wellman, 1999). However, this literature assumes that each expert provides a complete, self-contained probability distribution. This assumption is problematic, because human beings are notoriously poor at estimating probabilities or reasoning with them (Tversky & Kahneman, 1974). Our approach assumes a division of labor between humans and machines that better reflects their respective strengths and weaknesses (cf. Jaeger (2001)): humans provide qualitative statements that might be difficult for machines to discover unaided, and machines refine these statements and compute parameter estimates from data.

Our early results are quite promising. With both simulated and real experts, our method produces networks that are more accurate than the main alternatives: purely empirical learning, learning with knowledge from a single expert, and learning a separate model from each expert plus data and combining these.

The next section briefly reviews Bayesian network basics. We then describe our method and analyze it. This is followed by a section reporting our experimental results, and discussion.

2. Learning Bayesian Networks

A Bayesian network (Pearl, 1988) encodes the joint probability distribution of a set of d discrete variables, $\{x_1, \ldots, x_d\}$, as a directed acyclic graph and a set of conditional probability tables (CPTs). Each node corresponds to a variable, and the CPT associated with it contains the probability of each state of the variable given every possible combination of states of its parents. The set of parents of x_i, denoted $par(x_i)$, is the set of nodes with an arc to x_i in the graph. The structure of the network encodes the assertion that each node is conditionally independent of its nondescendants given its parents. Thus the probability of an arbitrary event $X = (x_1, \ldots, x_d)$ can be computed as $P(X) = \prod_{i=1}^{d} P(x_i|par(x_i))$.

When the structure of the network is known, learning reduces to estimating the CPT parameters. A widely-used algorithm for the case where the structure is unknown or imperfectly known is described by Heckerman et al. (1995). It performs a search over the space of network structures, starting from an initial network which may be random, empty, or derived from prior knowledge. At each step, the algorithm generates all variations of the current network that can be obtained by adding, deleting or reversing a single arc, without creating cycles, and selects the best one using the *Bayesian Dirichlet (BD)* score

$$\begin{aligned} P(S, D) &= P(S)P(D|S) \qquad (1) \\ &= P(S) \prod_{i=1}^{d} \prod_{j=1}^{q_i} \frac{\Gamma(n'_{ij})}{\Gamma(n'_{ij} + n_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(n'_{ijk} + n_{ijk})}{\Gamma(n'_{ijk})} \end{aligned}$$

where S is a network structure, D is a training set of n iid examples without missing values, $\Gamma()$ is the gamma function, q_i is the number of states of the Cartesian product of x_i's parents, r_i is the number of states of x_i, n_{ijk} is the number of occurrences of the kth state of x_i with the jth state of its parents, and $n_{ij} = \sum_{k=1}^{r_i} n_{ijk}$. $P(S)$ is the prior probability of the structure, which Heckerman et al. set to an exponentially decreasing function of the number of different arcs in S and the initial (prior) network. Each multinomial distribution for x_i given a state of its parents has an associated Dirichlet prior distribution with parameters n'_{ijk}, with $n'_{ij} = \sum_{k=1}^{r_i} n'_{ijk}$. These parameters can be thought of as equivalent to seeing n'_{ijk} occurrences of the corresponding states in advance of the training examples. The BD score is the result of integrating over the resulting posterior distribution for the parameters of each multinomial. The search ends, and returns the current network, when no variation achieves a higher BD score. See Heckerman et al. (1995) for more details on learning Bayesian networks.

This paper proposes and evaluates a method for finding the structure of a Bayesian network from a multitude of expert statements about it, or more generally for computing the prior distribution $P(S)$ over possible structures.

3. Approach

Our approach is summarized in Figure 1. The world, assumed modelable by some Bayesian network, generates both data and the imperfect knowledge of m experts about the structure of the network. The expert knowledge is used to compute a probability distribution over structures. Given the most probable structure and the data, we learn the parameters of the network. Optionally, given the data and the expert-induced distribution over structures, the learner finds the *a posteriori* most probable structure and the posterior distribution over parameters. (Ideally, the learner would average over all structures, but this is computationally infeasible.)

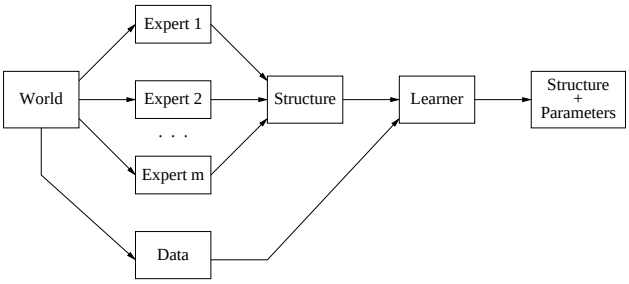

Figure 1. Learning with knowledge from multiple experts.

3.1. Basic Framework

We represent the structure of a Bayesian network with d nodes as a vector $S = (s_1, \ldots, s_j, \ldots, s_{d'})$ of $d' = d(d-1)/2$ *structure variables*, where s_j corresponds to the jth node pair in some arbitrary ordering of all the possible node pairs. For any two nodes X and Y, only one of the two pairs (X, Y) and (Y, X) is considered, the choice of which being arbitrary. Pairs of the form (X, X) are not considered. If s_j corresponds to the pair (X, Y), then $s_j = \emptyset$ if there is no arc between X and Y in the network, $s_j = \rightarrow$ if there is an arc from X to Y, $s_j = \leftarrow$ if there is an arc from Y to X, and $s_j = \leftrightarrow$ if there is an arc from to X to Y and an arc from Y to X. (Although the latter cannot happen in a real network, it may be stated by a noisy expert, and therefore we need to allow for it.)

We assume we have a pool of m experts, and the ith expert provides a vector $E_i = (e_{i,1}, \ldots, e_{ij}, \ldots, e_{id'})$, where $e_{ij} \in \{\emptyset, \rightarrow, \leftarrow, \leftrightarrow\}$ states the expert's belief about the existence of an arc between the nodes in the jth pair. Thus all the expert knowledge available is contained in the matrix $E = (E_1, \ldots, E_i, \ldots, E_m)$. In other words, each expert tells us what s/he believes are the dependencies in the domain. Because an expert's knowledge of the world is imperfect, E_i is a noisy version of S. We will often slightly abuse notation and consider E_i to "be" the ith expert.

A Bayesian network is composed of a structure S and a parameter vector Θ. Our goal is to induce a posterior distribution over (S, Θ) given expert knowledge E and a training set D. This distribution can then be used to find the most probable network (S^*, Θ^*), or to compute probabilities of interest by averaging over the possible networks, weighted by their posteriors. To make the problem tractable, we introduce a number of simplifying assumptions. By Bayes' theorem,

$$\begin{aligned} & P(S, \Theta | E, D) \\ &= \alpha P(S, \Theta) P(E, D | S, \Theta) \\ &= \alpha P(S) P(\Theta | S) P(E | S, \Theta) P(D | E, S, \Theta) \end{aligned} \quad (2)$$

We use α throughout to represent a normalizing constant (not necessarily always the same one). We assume that $P(E|S, \Theta) = P(E|S)$ (i.e., expert statements about structure depend only on the structure, not the parameters) and that $P(D|E, S, \Theta) = P(D|S, \Theta)$ (i.e., the data is independent of the experts given the actual structure and parameters). Substituting these equalities into Equation 2 and integrating both sides over Θ yields the posterior over structures

$$\begin{aligned} P(S|E, D) &= \alpha P(E|S) P(S, D) \\ &= \alpha P(S) P(E|S) P(D|S) \end{aligned} \quad (3)$$

where $P(S, D)$ is given by Equation 1. The quantity $P(E|S)P(S, D)$ is the new BD score, extended to take expert statements into account by replacing the prior $P(S)$ with the "post-expert" prior $P(S|E) = \alpha P(S)P(E|S)$ (i.e., the distribution over structures after consulting the experts but before seeing any data). It can be used as the scoring function in any algorithm that learns a Bayesian network by searching over structures.

3.2. Expert Model

$P(E|S)$ is the generative model for expert statements, and is the key new quantity we introduce. It can be represented by a "meta-level" Bayesian network with nodes $\{s_1, \ldots, s_{d'}, e_{1,1}, \ldots, e_{1,d'}, \ldots, e_{m,1}, \ldots, e_{md'}\}$, where the s_i nodes have known values and no parents. Thus, if $par(e_{ij})$ denotes the parents of node e_{ij},

$$P(E|S) = \prod_{i=1}^{m} \prod_{j=1}^{d'} P(e_{ij} | par(e_{ij})) \quad (4)$$

In this paper we assume the simplest useful case, which is that $par(e_{ij}) = \{s_j\}$. In other words, an expert's probability of stating that an arc exists depends only on whether the actual arc or its reverse exist, and not on what other experts state, or what the expert states about other arcs. In particular, this implies that the experts are independent given the actual structure (i.e., that they "distort reality" in independent ways; note that this is is quite different from the experts being unconditionally independent). These assumptions obviously oversimplify the behavior of real experts, but may lead to better performance than more realistic ones whose parameters would be hard to estimate.

The expert model is thus fully specified by specifying $P(e_{ij}|s_i)$ for all (i, j). We assign the same *a priori* values to these parameters for all (i, j).[1] The natural

[1]The *a posteriori* estimates, after we have seen E, can be different for different (i, j). However, making the usual parameter independence assumption (Heckerman et al., 1995), observation of one e_{ij} does not affect estimation of any other. (Notice that, in our treatment, E is composed of exactly one sample of each e_{ij} variable.)

Table 1. Parameters of the expert model, $P(e_{ij}|s_j)$.

s_j	e_{ij}			
	$\emptyset$	$\rightarrow$	$\leftarrow$	$\leftrightarrow$
$\emptyset$	$1-2p_a-p_b$	p_a	p_a	p_b
$\rightarrow$	p_d	$1-p_d-p_r-p_c$	p_r	p_c
$\leftarrow$	p_d	p_r	$1-p_d-p_r-p_c$	p_c

parameters to state $P(e_{ij}|s_j)$ in terms of are: p_a, the probability that the expert adds an arc in a particular direction where there was none; p_d, the probability that the expert deletes an arc; p_r, the probability that the expert reverses an arc; p_b, the probability that the expert creates a cycle (arcs in both directions) where there was no arc; and p_c, the probability that the expert creates a cycle where there was an arc. The resulting values of $P(e_{ij}|s_j)$ for all (i, j) are shown in Table 1. (Since $s_i = \leftrightarrow$ cannot occur, it is not necessary to specify $P(e_{ij}|s_j)$ for this case.) A simple and possibly very useful extension is to have different values of the parameters for each expert (i.e., $P(e_{ij}|s_j) = p_{ai}$, etc.).

Notice that there is nothing to preclude an expert from specifying a structure with cycles. Even if incorrect, such a structure will in general still contain useful information, and our method allows us to extract it.

We also need to specify a prior $P(S)$ over structures. In this paper we assume that each pair of nodes independently has some probability p_0 of being connected by an arc in a given direction. We also know that Bayesian networks must be acyclic, and so set $P(S) = 0$ for any S containing cycles:

$$P(S) = \alpha C(S) \prod_{j=1}^{d'} P(s_j) \tag{5}$$

where $C(S) = 0$ if S contains a cycle and $C(S) = 1$ otherwise, and $P(s_j) = 1 - 2p_0$ if $S_j = \emptyset$, $P(s_j) = p_0$ if $S_j = \rightarrow$ or $S_j = \leftarrow$, and $P(s_j) = 0$ if $S_j = \leftrightarrow$. Combining Equations 4 and 5 yields

$$\begin{aligned} P(S|E) &= \alpha P(S)P(E|S) \\ &= \alpha C(S) \prod_{j=1}^{d'} P(s_j) \prod_{i=1}^{m} P(e_{ij}|s_j) \end{aligned} \tag{6}$$

3.3. Analysis

We now derive an expression for the probability that our method incorrectly predicts the value of a structure variable s_j, assuming that the expert model and parameters it uses are correct. Viewed another way, this is the expected fraction of incorrect arcs in the structure predicted by the ensemble of experts. More precisely, if $\hat{s}_j$ is the value of s_j with highest $p_j = P(s_j) \prod_{i=1}^{m} P(e_{ij}|s_j)$ (see Equation 6), the expression

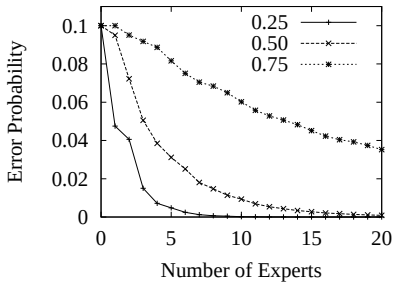

Figure 2. Error probability as a function of the number of experts and their noise level.

is for the probability that $\hat{s}_j \neq s_j$ (i.e., for the probability of making an incorrect prediction before breaking cycles). $P(\hat{s}_j \neq s_j)$ can be expanded as follows:

$$P(\hat{s}_j \neq s_j) = \sum_{v_j \in \{\emptyset, \rightarrow, \leftarrow, \leftrightarrow\}} P(s_j = v_j) P(\hat{s}_j \neq s_j | s_j = v_j) \tag{7}$$

In turn, $P(\hat{s}_j \neq s_j | s_j = v_j)$ can be expanded thus:

$$P(\hat{s}_j \neq s_j | s_j = v_j) = \sum_{\mathcal{E}} P(\hat{s}_j \neq s_j | e_{1:m,j}, s_j = v_j) \prod_{i=1}^{m} P(e_{ij}|s_j = v_j) \tag{8}$$

where the sum is over the set $\mathcal{E}$ of all 4^m possible vectors of expert statements $e_{1:m,j} = (e_{1,j}, \ldots, e_{mj})$ about s_j. The probabilities $P(e_{ij}|s_j = v_j)$ can be obtained from Table 1. $P(\hat{s}_j \neq s_j | e_{1:m,j}, s_j = v_j) = 0$ if p_j is higher for v_j than for any other value (i.e., the correct prediction is made), and $P(\hat{s}_j \neq s_j | e_{1:m,j}, s_j = v_j) = 1$ otherwise. Replacing this into Equation 8 and the latter into Equation 7, we obtain a long but straightforward expression for the error probability $P(\hat{s}_j \neq s_j)$. (The number of terms in Equation 8 can be greatly reduced by counting and combining all terms with the same number of experts predicting each value of s_j.) This probability as a function of the number of experts and the noise level is plotted in Figure 2. A noise level of ρ is defined as the expert parameter settings in which the expert removes or reverses a fraction ρ of the arcs, and adds the same number between unrelated nodes. We expect edge deletion to be much more frequent in practice than edge reversal, so we let $p_d = 4p_r$. We set the prior p_0 to 0.05. As can be seen, the error probability is low even with few experts and a substantial noise level. See also the analyses of model ensembles in Hansen and Salamon (1990) and Perrone and Cooper (1993).

3.4. Algorithm

We use the hill-climbing search algorithm of Heckerman et al. (1995) to find the best structure, initializing it with the structure that maximizes $P(S|E)$, and replacing $P(S)$ by $P(S|E)$ in the BD score (Equation 1),

as described in the previous section. However, learning the structure of a Bayesian network can be computationally expensive. A common alternative, if the expert knowledge is of sufficient quality, is to take the structure given by the expert(s) as the "true" one, and simply fit parameters to this structure. In our context, this means taking the structure with highest $P(S|E)$ as the "true" one, and learning parameters. We also explore this alternative in our experiments.

Consider Equation 6, and ignore for the moment the problem of cycles (i.e., the $C(S)$ term). Because the term for each structure variable s_j is independent of all others, the structure with highest $P(S|E)$ can be found in $O(d^2m)$ time simply by setting each s_j to the value with highest $p_j = P(s_j)\prod_{i=1}^{m} P(e_{ij}|s_j)$. If we assume that the structures provided by the experts have some sparse number of arcs $e \ll d^2$, and the expert parameters $(p_a, p_d, \ldots)$ are set reasonably so that an arc can only be introduced between a pair of nodes j if $\exists i\ s_{ij} \neq \emptyset$, then this computation takes only $O(em)$ time. However, the structure S' found in this way may contain cycles. We thus heuristically find the structure with highest $P(S|E)$ by breaking all cycles in S' while minimally reducing its score $\prod p_j$. We use the procedure of Hulten et al. (2003), which finds the set of arcs involved in cycles by finding the graph's strongly connected components, and breaks cycles by greedily removing the component arcs with lowest p_j. Though not guaranteed to find the acyclic structure with highest probability, in our experience this quickly finds an acyclic structure using very few arc removals.

We set the parameters p_a, p_d, p_r, p_b, p_c to optimize log-likelihood, measured by two-fold cross-validation. Optimization is done either by sequentially trying a predetermined set of values for each parameter, or by Powell's method (Press et al., 1992).

A further issue that arises is that the available data may be insufficient to reliably fit the parameters of the structure with highest $P(S|E)$, leading to poor performance, even if this structure is the "correct" one. In this case an oversimplified structure (with fewer parents per node, and thus more data for each CPT entry) might perform better. We address this issue by performing *shrinkage* of the parameters as follows. For each node x_i, we order the parents $par(x_i)$ as follows: $par_1(x_i)$ is the parent with highest mutual information with respect to x_i; $par_h(x_i)$ is the parent with highest mutual information given $\{par_1(x_i), \ldots, par_{h-1}(x_i)\}$. Let $\hat{p}_{ijk0}$ be the unshrunken estimate of $p_{ijk} = P(x_i = k \mid par(x_i) = j)$ (i.e., conditioning on all parents). Let $\hat{p}_{ijks}$ be the estimate obtained by ignoring the last s parents in the ordering. The shrunken estimate is then $\hat{p}_{ijk} = \sum_{s=0}^{|par(x_i)|} \lambda_s \hat{p}_{ijks}$. The shrinkage coefficients λ are found using the EM algorithm, as described in McCallum et al. (1998). The shrinkage is incorporated into the parameter estimation and structure search by appropriately setting the Dirichlet parameters (see Equation 1): $n'_{ijk} = (n_{ij}/\lambda_0)\sum_{s=1}^{|par(x_i)|} \lambda_s p_{ijks} + 1$.

Table 2. Network characteristics and results. p_0^* is the true probability of an arc between two nodes. δ_i is the reduction in K-L distance achieved when using i experts, as a fraction of the maximum possible (difference between learning with the empty network and learning with the the true one).

Network	Nodes	Edges	p_0^*	δ_1	δ_{10}
Alarm	37	46	0.035	47%	93%
Hailfinder	56	66	0.021	55%	98%
Insurance	27	52	0.074	57%	98%
Water	32	66	0.067	51%	98%

4. Experiments

We performed two sets of experiments. In the first, we used simulated experts and data generated from benchmark Bayesian networks to study the effect of training set size, number of experts, and noise level on our algorithm. In the second, we used knowledge of printer troubleshooting from nine computer users, and data from the Microsoft Windows printer diagnosis network. In both cases, the domains used are considerably simpler than those we envisage ultimately applying our approach to, being generated from known simple Bayesian networks. Nevertheless, these experiments should be useful as preliminary tests.

4.1. Simulated Experts

We used four networks from the Bayesian network repository at http://www.cs.huji.ac.il/labs/compbio/Repository/ as our ground truths. Table 2 lists the networks and some of their characteristics. Since the correct prior probability of an edge, p_0^*, is unknown in practice, we set p_0 to be 0.05 for all networks. Expert parameters were set as in Section 3.3. Unless specified, the default noise level is 0.5 ($p_a = 0.025$, $p_d = 0.4$, $p_r = 0.1$, $p_c = 0$, $p_b = 0$), and the training set size is 100 examples.

We generated expert statements from the true networks according to the model described in the previous section. The parameters for the structure inference algorithm were not set to the true values, but optimized as they would need to be in a real-world situation.[2] After optionally performing a search over

[2]Sequential search performed slightly better than Powell's method, and is the one we report. Both methods take negligible time compared to the structure learning.

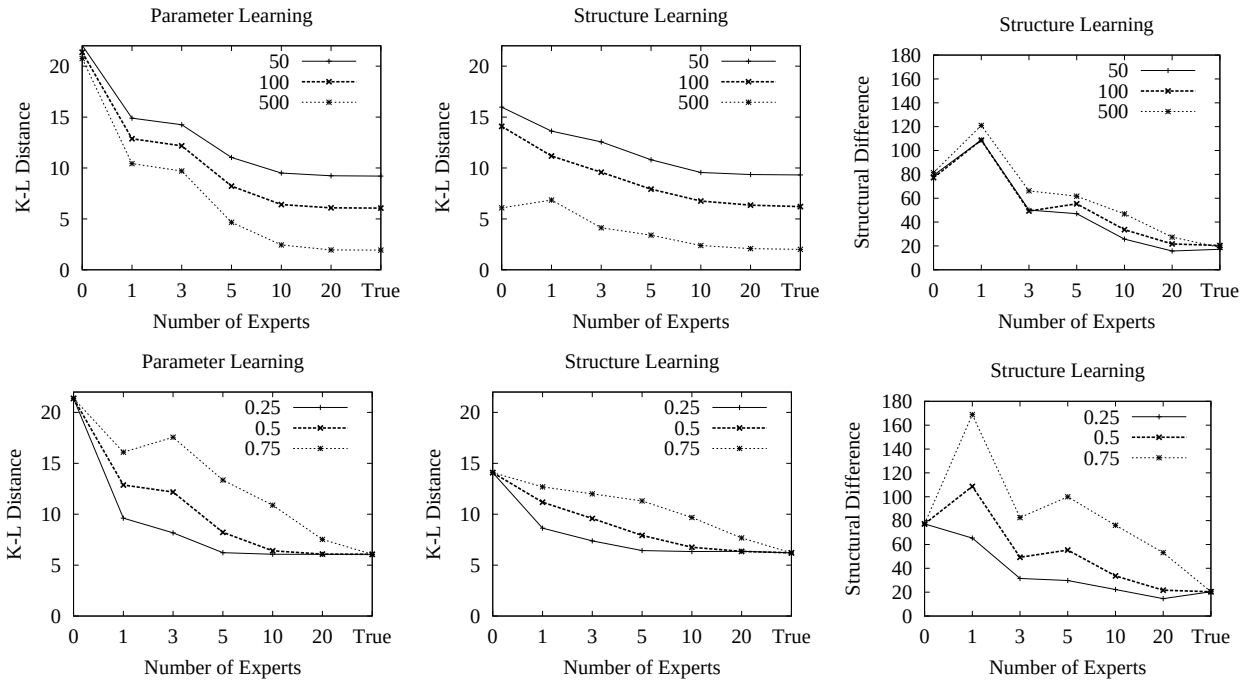

Figure 3. Experimental results for simulated experts: varying training set size (top) and varying noise level (bottom). The left-hand graphs were obtained by finding the best structure using our method with 3 to 20 experts, and estimating parameters for it. "0" is the empty structure, "1" is a single expert's structure (with cycles removed), and "True" is the true structure. The middle and right-hand graphs were obtained by finding the initial structure and prior over structures using our method with 3 to 20 experts, and applying Heckerman et al.'s (1995) structure-learning algorithm. In "0" the initial structure is empty, in "1" it is a single expert's structure, and in "True" it is the true structure; in these three cases, Heckerman et al.'s algorithm was applied with their prior. In all cases, network parameters were learned with shrinkage.

structures (guided by $P(S|E, D)$), the resulting network was evaluated using two measures: a) the average K-L distance from the true network, estimated using 100k samples,[3] and b) the structural difference between the learned network and the true one, measured as the number of arcs added or removed, with reversals counting as two differences.

We compared expert combination with three other cases: *zero experts* (purely empirical learning, starting from an empty network), *one expert* (the network provided by one expert), and *true* (purely empirical learning, starting from the true network). For these we used the same prior $P(S)$ as in Heckerman et al. (1995), which is a discount of a factor of κ for every structural change from the initial model. We searched for the best κ in $\{0.01, 0.1, 0.2, \ldots, 0.8, 0.9, 0.99\}$ using two-fold cross-validation.[4] Results are averaged over 20 runs; each has independent training sets and experts, but all share the test set. Results for each run were on varying subsets of a fixed set of 20 experts. Within each run, each successively larger set of experts was obtained by adding randomly-chosen experts to the previous (smaller) set.

Results were qualitatively similar in all four domains. Table 2 shows summary results for parameter learning in the four domains. A single expert is useful, but still quite error-prone. In contrast, ten experts are sufficient to essentially recover the true network. Full results for hailfinder, the most complex domain, are shown in Figure 3.[5] With or without structure learning, multiple experts systematically outperform a single expert, as well as purely empirical learning, in both K-L distance and structural difference. This illustrates the potential of our approach. A single expert outperforms structure learning in K-L distance but not in structural difference, suggesting that using multiple experts is particularly important when the goal is to understand the structure of the domain, rather than just obtain accurate predictions.

[3]This is equivalent to the negative log-likelihood on these samples, minus the entropy of the true distribution estimated from the samples.

[4]Notice that the cross-validation for κ was performed with structure search, while for the parameters of our expert model it was only performed with parameter learning, the results being used for structure learning as well.

[5]Notice K-L distances for "true" are not zero, because they refer to *learning* starting from the true structure.

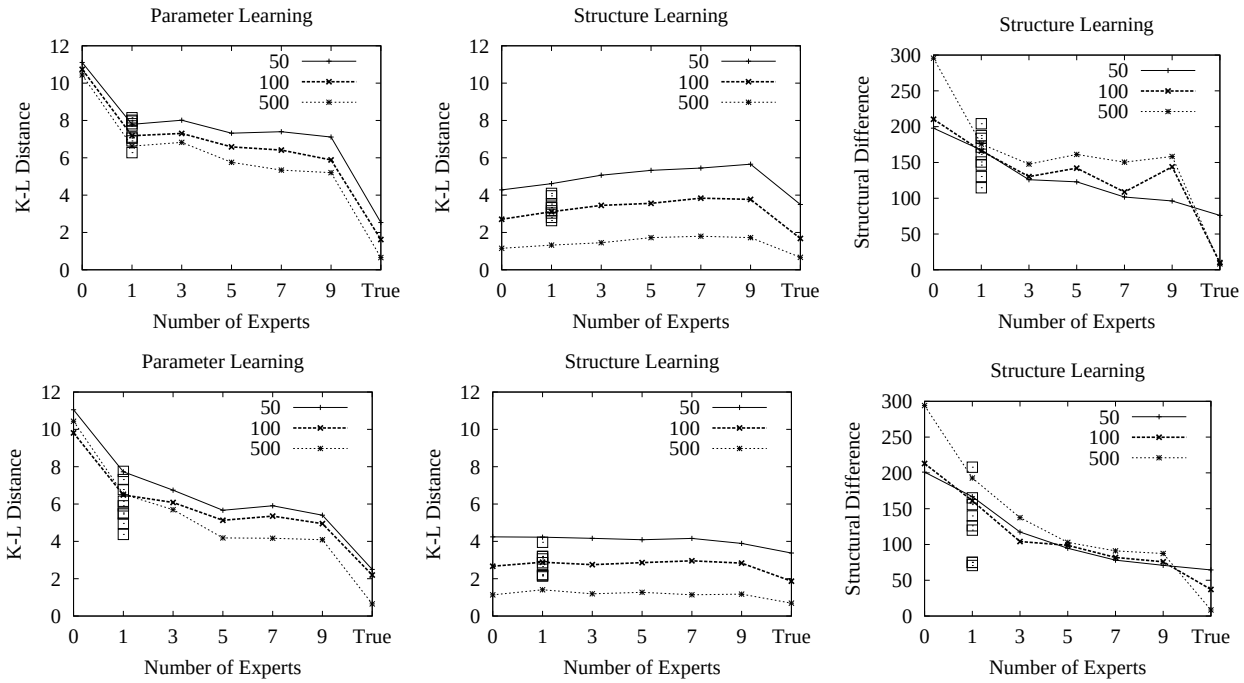

Figure 4. Experimental results in printer domain: low expertise (top) and high expertise (bottom). Please see the previous figure's caption for details of the experimental procedure.

4.2. Real Experts

One focus of our collective knowledge base project has been the development of a computer troubleshooting system. A significant number of Usenet newsgroups, FAQs, and discussion forums are devoted to the task of helping others diagnose their computer problems, which suggests that a collective knowledge base for this domain would be in high demand, and that experts may be readily found on the Internet to contribute to such a knowledge base. As a simple simulation of a subset of this domain, we used the Microsoft printer troubleshooting Bayesian network (also in the Bayesian network repository). This network has 76 nodes and 112 edges, and $p_0^* = 0.02$.

We used the same experimental procedure as in the previous section. Our "experts" were nine computer users who were familiar with printing but not intimately knowledgeable about the details of Windows printing. Each expert was given a list of the domain variables and asked to list the direct causes of each. To simulate more knowledgeable experts, we let each expert then study the true structure of the network for a period of time, asking them to try to understand the reason why certain variables were causes of other variables (we assisted in this task by providing an explanation for each).[6] After this, the experts were asked again to list the causes of each variable. We call these two cases "low-expertise" and "high-expertise" respectively. The experts spent from two to five hours on the task. We set $p_0 = 0.02$, and tried optimizing the parameters of the expert model both globally and separately for each expert. The latter alternative (with Powell's method) performed slightly better, and is the one we report.

The results, averaged over 20 runs, are shown in Figure 4. In each run, the subset of experts used for each number of experts was randomly chosen. The boxes represent the scores of the individual experts with 100 examples, averaged across runs. Without structure learning, experts always produced better networks than the purely empirical approach, and multiple experts combined using our method outperformed using a single one, on average. With structure learning, all methods performed similarly in K-L distance. This can be seen to be due to the lack of room for improvement between purely empirical learning and starting with the true network, and can be attributed to the simplicity of the domain. We expect the benefits of our approach to be more visible on large, complex domains where the data is generated by the real world, instead of by a model network. Experts outperformed purely empirical learning, and multiple experts outperformed one, in structural difference, showing the potential advantages of our method for obtaining insight.

[6]The experts' instructions are at http://www.cs.washington.edu/homes/pedrod/lkme-task.pdf.

The average K-L distance of the boxes (in the left and middle plots) is the K-L distance that would result from combining expert predictions with a logarithmic pool (French, 1985) (i.e., from combining the models that would result from fitting parameters separately to each expert structure). Without structure learning, the combined experts outperformed the logarithmic pool, and all but the best of the individual experts. This shows that our method can be preferable to standard model combination techniques. With structure learning, the K-L distances of our method and the logarithmic pool were similar, for the reasons described above. Even in this case, our method may still be preferable, because it produces a single comprehensible structure, in contrast to the multiple structures maintained by *a posteriori* combination methods.

Even on these simple networks, structure learning was an order of magnitude slower than parameter learning. In domains with many thousands of variables, where expertise combination is likely to be most useful, structure learning may simply not be feasible. Our method with only parameter learning is likely to be the best choice in this case. Given that structure learning is likely to be faster when starting closer to the true network, there may also be cases where our method with structure learning is feasible, while purely empirical structure learning is not.

5. Conclusions and Future Work

This paper proposes that knowledge-intensive learning can be facilitated by allowing the expert input to be noisier, and making up for this by using multiple experts. In particular, we develop a method for combining a multitude of (possibly inconsistent) statements about the structure of a Bayesian network into a "best guess" about the actual structure. Empirical data is then used to refine and parameterize this structure. Preliminary experiments with real and simulated experts indicate that this approach is indeed useful.

Directions for future research include: supporting more varied types of input from experts; using more refined models of expert behavior; applying our approach to other representations besides Bayesian networks; testing it in real domains where the ground truth is unknown; combining it with information extraction techniques to exploit redundancy in online sources; and incorporating it into our Web-based system for collecting and refining expert knowledge.

Acknowledgements: We are grateful to David Heckerman for helpful discussions. This research was partly supported by an IBM Ph.D. Fellowship to the first author, and by ONR grant N00014-02-1-0408.

References

Bergadano, F., & Giordana, A. (1988). Knowledge-intensive concept induction. *ICML-88* (pp. 305–317).

Breiman, L. (1996). Bagging predictors. *Mach. Learn.*, *24*.

Frauenfelder, M. (2000). Revenge of the know-it-alls: Inside the Web's free-advice revolution. *Wired*, *8*, 144–158.

French, S. (1985). Group consensus probability distributions. In *Bayesian statistics 2*, 183–202. Elsevier.

Freund, Y., & Schapire, R. E. (1996). Experiments with a new boosting algorithm. *ICML-96* (pp. 148–156).

Genest, C., & Zidek, J. V. (1986). Combining probability distributions. *Statistical Science*, *1*, 114–148.

Hansen, L. K., & Salamon, P. (1990). Neural network ensembles. *IEEE Trans. PAMI*, *12*, 993–1000.

Heckerman, D., Geiger, D., & Chickering, D. M. (1995). Learning Bayesian networks. *Mach. Learn.*, *20*, 197–243.

Henrion, M. (1987). Some practical issues in constructing belief networks. *UAI-87* (pp. 161–173).

Hulten, G., Chickering, D. M., & Heckerman, D. (2003). Learning Bayesian networks from dependency networks. *AIStats-03*.

Jaeger, M. (2001). Constraints as data: A new perspective on inferring probabilities. *IJCAI-01* (pp. 755–760).

Lenat, D. B., & Guha, R. V. (1990). *Building large knowledge-based systems*. Addison-Wesley.

Lindley, D. V. (1985). Reconciliation of discrete probability distributions. In *Bayesian statistics 2*, 375–390. Elsevier.

Marcus, S. (ed.) (1989). Special issue on knowledge acquisition. *Mach. Learn.*, *4*.

McCallum, A., Rosenfeld, R., Mitchell, T., & Ng, A. Y. (1998). Improving text classification by shrinkage in a hierarchy of classes. *ICML-98* (pp. 359–367).

Morris, J. (August 2001). CRM leaders: Compaq gets it right. *Customer Support Management*.

Ourston, D., & Mooney, R. J. (1994). Theory refinement combining analytical and empirical methods. *Artif. Intel.*, *66*, 273–309.

Pazzani, M., & Kibler, D. (1992). The utility of knowledge in inductive learning. *Mach. Learn.*, *9*, 57–94.

Pazzani, M., Mani, S., & Shankle, W. R. (1997). Comprehensible knowledge discovery in databases. *CogSci-97*.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems*. Morgan Kaufmann.

Pennock, D., & Wellman, M. (1999). Graphical representations of consensus belief. *UAI-99* (pp. 531–540).

Pennock, D. M., Nielsen, F. A., & Giles, C. L. (2001). Extracting collective probabilistic forecasts from Web games. *KDD-2001* (pp. 174–183).

Perrone, M. P., & Cooper, L. M. (1993). Ensemble methods for hybrid neural networks. In *Artificial neural networks for speech and vision*, 126–142. Chapman and Hall.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (1992). *Numerical recipes in C* (2nd ed.). Cambridge University Press.

Richardson, M., & Domingos, P. (2003). Building large knowledge bases by mass collaboration. Tech. Rept. 03-02-04, Dept. CSE, Univ. Washington, Seattle, WA.

Scott, A., Clayton, J., & Gibson, E.. (1991). *A practical guide to knowledge acquisition*. Addison-Wesley.

Stork, D. G. (2000). Using open data collection for intelligent software. *IEEE Computer*, *33*, 104–106.

Towell, G. G., & Shavlik, J. W. (1994). Knowledge-based artificial neural networks. *Artif. Intel.*, *70*, 119–165.

Tversky, A., & Kahneman, D. (1974). Judgment under uncertainty. *Science*, *185*, 1124–1131.

Wolpert, D. (1992). Stacked generalization. *Neural Networks*, *5*, 241–259.