Learning Accurate Belief Nets

Wei Zhou and Russell Greiner
Department of Computing Science
615 General Service Bldg, University of Alberta
Edmonton, AB T6G 2H1 Canada
{ wei, greiner }@cs.ualberta.ca

Abstract

Bayesian belief nets (BNs) are typically used to answer a range of queries, where each answer requires computing the probability of a particular hypothesis given some specified evidence. An effective BN-learning algorithm should, therefore, learn an *accurate* BN, which returns the correct answers to these specific queries. This report first motivates this objective, arguing that it makes effective use of the data that is encountered, and that it can be more appropriate than the typical "maximum likelihood" algorithms for learning BNs. We then describe several different learning situations, which differ based on how the query information is presented. Based on our analysis of the inherent complexity of these tasks, we define three algorithms for learning the best CPtables for a given BN-structure, and then demonstrate empirically that these algorithms work effectively.

1 Introduction

Many tasks require answering questions; this model applies, for example, to both expert systems that identify the underlying fault from a given set of symptoms, and control systems that propose actions on the basis of sensor readings. When the mapping from evidence to fault/response is stochastic, it makes sense to represent it using a probabilistic model. One obvious candidate is a "(Bayesian) belief net" (BN), which succinctly encodes a distribution over a set of variables.

Often the underlying distribution, which is needed to map questions to appropriate responses, is not known *a priori*. In such cases, if we have access to training examples, we can try to *learn* the model. Our goal, of course, is an *accurate* BN — *i.e.*, which returns the correct answer as often as possible. While there are many algorithms that learn BNs from data [Hec95, Bun96], most of these systems attempt to maximize another criteria — typically some variant of likelihood — which is independent of the queries posed.

While a perfect model of the distribution will perform optimally over all possible queries, a learner cannot guarantee producing such a model without seeing a very large number of samples. As most learners have access to only a limited set of samples, it is important that they use these samples effectively. Here, this means focusing on those aspects of the distribution that will contribute most to the accuracy.

For example, if we know that all of the queries will ask for the probability of some specific disease (drawn from a specified set), given a set of symptoms (read "specific assignments to specified variables"), we clearly want our learner to produce a BN B_Q that answers these questions correctly, even if this B_Q would produce completely wrong answers to other (unasked) questions — *e.g.*, about the dependencies of one symptom on another.

This paper therefore continues the argument, begun in [GGS97], that BN-learning algorithms should consider the distribution of *queries*, as well as the underlying distribution of events, and should seek the BN with the best performance *over the query distribution*. Below we investigate the challenges of learning such accurate BNs. Section 2 first overviews belief nets, then presents the three specific learning models we are considering, which vary depending on what the learner is given — *i.e.*, what type of "training data" is available. The next three sections explore these three models, first using formal analyses (*e.g.*, about the hardness of various tasks) to motivate our specific leaning algorithms, and then reporting the results obtained by our implementations of such algorithms. Throughout, we argue, both theoretically and empirically, that our models often improve on the traditional (maximizing likelihood) models and their associated algorithms.

2 Framework

2.1 Overview of Belief Nets

In general, we assume there is a stationary underlying distribution $P(\cdot)$ over the N (discrete) variables $\mathcal{V} = \{V_1, \ldots, V_N\}$. For example, perhaps V_1 is the "`Cancer`" random variable, whose value ranges over $\{$ `true`, `false` $\}$; V_2 is "`Gender`" $\in$ {`male`, `female`}, V_3 is "`Age`" $\in [0..100]$, etc. We will refer to this as the "underlying distribution" or the "tuple distribution".

We can encode this as a "(Bayesian) belief net" (BN; a.k.a. Bayesian network, probability net, causal net), which is a directed acyclic graph $B = \langle \mathcal{V}, E, \Theta \rangle$, whose nodes $\mathcal{V}$ represent variables, and whose arcs E represent dependencies. Each node $V_i \in \mathcal{V}$ also includes a conditional-probability-table (CPtable) $\theta_i \in \Theta$, that specifies how V_i's values depends (stochastically) on the values of its parents. (Readers unfamiliar with these ideas are referred to [Pea88].)

User interact with the belief net by asking *queries*, each being a term of the form "$P(\mathbf{X} = \mathbf{x} \,|\, \mathbf{Y} = \mathbf{y}) = ?$" — *e.g.*, $P($`Cancer = true` $|$ `Gender=female, Age=35, Smoke=true` $)$ — where $\mathbf{X}, \mathbf{Y} \subset \mathcal{V}$ are subsets of $\mathcal{V}$, and $\mathbf{x}$ (resp., $\mathbf{y}$) is a legal assignment to the elements of $\mathbf{X}$ (resp., $\mathbf{Y}$). (We of course allow $\mathbf{Y} = \{\}$.) The numeric value of each such query, called its "label", is the posterior probability value of this (conditional) event, over the underlying distribution. For example, "0.65" is the label of the labeled query "$P(\text{cancer} \,|\, \text{female, 35yo, smoker}) = 0.65$".

We let $\mathcal{SQ}$ be the set of all possible legal statistical queries, and assume there is a (stationary) distribution over $\mathcal{SQ}$, written $sq(\mathbf{X} = \mathbf{x}\,;\, \mathbf{Y} = \mathbf{y})$, where $sq(\mathbf{X} = \mathbf{x}\,;\, \mathbf{Y} = \mathbf{y})$ is the probability that the query "What is the value of $P(\mathbf{X} = \mathbf{x} \,|\, \mathbf{Y} = \mathbf{y})$?" will be asked. To simplify our notation, we will often use a single variable, say q, to represent the $[\mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y}]$ query, and so will write $sq(q)$ to refer to $sq(\mathbf{X} = \mathbf{x}\,;\, \mathbf{Y} = \mathbf{y})$. We will also write a "labeled query" as $\langle \mathbf{X} = \mathbf{x};\ \mathbf{Y} = \mathbf{y};\ p \rangle$, where $p = P(\mathbf{X} = \mathbf{x} \,|\, \mathbf{Y} = \mathbf{y})$.[1]

Notice the query distribution ($sq(\cdot)$) can be completely unrelated to the underlying distribution (P) — *e.g.*, even though *"What is $P($Cancer$\,|\,$female, 35yo, smoker$)$?"* is asked 35% of the time, the actual value of $P(\text{Cancer} \,|\, \text{female, 35yo, smoker})$ could be 0, or 1, or any other value; see [GGS97].

[1] A query $sq(\mathbf{X} = \mathbf{x}\,;\, \mathbf{Y} = \mathbf{y})$ is "legal" if $P(\mathbf{Y} = \mathbf{y}) > 0$. Note also that we use *CAPITAL* letters to represent single variables, *lowercase* letters for the values that the variables might assume, and the **boldface** font when dealing with *sets* of variables or values. Also, within the query $P(\mathbf{X} = \mathbf{x} \,|\, \mathbf{Y} = \mathbf{y})$ we refer to $\mathbf{X}$ as the "query variable(s)" and $\mathbf{Y}$ as the "evidence variable(s)". We also write $sq(\mathbf{x}\,;\, \mathbf{y})$ to refer to the probability of the query "$P(\mathbf{X} = \mathbf{x} \,|\, \mathbf{Y} = \mathbf{y}) = ?$" where the variable sets $\mathbf{X}$ and $\mathbf{Y}$ can be inferred from the context.

```
⟨ Cancer = t;   Gend=F, Age=35, Smoke=t;        0.65  ⟩
⟨ Cancer = t;   Gend=M, Age=25, Smoke=f;        0.001 ⟩
⟨ Cancer = f;   LivBio=true;                    0.3   ⟩
⟨ Menin = t;    Gend=F, BTest=true;             0.71  ⟩
⟨ Menin = t;    Gend=F, LivBio=f, Temp=High;    0.002 ⟩
```

Figure 1: Sample of Labeled Queries

2.2 Learning Contexts and Algorithms

As motivated above, the goal of our learning system is to produce a performance system (read "belief net") that can return the appropriate answers to queries from this distribution. We describe below three different learners, each designed to accommodate a different class of training information. In all cases, we will focus on the task of learning the CPtables for a given BN-structure.

2.2.1 Explicitly-Labeled Queries

The most generous environment provides an explicit set of "labeled queries" to the learner: That is, the learner sees data of the form $\mathcal{LQ} = \{\langle \mathbf{X}_i = \mathbf{x}_i,\ \mathbf{Y}_i = \mathbf{y}_i,\ p_i \rangle\}_{i=1..N}$, where $p_i \in [0,1]$ is the label of the query "$P(\mathbf{X}_i = \mathbf{x}_i \,|\, \mathbf{Y}_i = \mathbf{y}_i)$". More precisely, we assume these $[\mathbf{X}_i = \mathbf{x}_i,\ \mathbf{Y}_i = \mathbf{y}_i]$ queries are drawn at random from the query distribution $\mathcal{SQ}$, then each such query is "labeled" based on the underlying probability P. To illustrate the ideas, Figure 1 displays a small sample of labeled queries, S_{LQ}. Notice different queries can involve different query variables, as well as different types of evidence variables. note also that this "labeled query sample" does not include every possible query, as it does not include other combinations of test results: it omits `cancer` given `Gender=Female, Age=35, Smoke=false`, and many other combinations of values for these specific evidence variables (but see "query forms" below), omits `cancer` given other sets of symptoms, and omits many other query/evidence combinations; *e.g.*, it never deals with `BloodTest` given `Fever`, nor with `Meningitis` given `Cancer`, etc. Note that many of these possible queries would not occur even in larger (or even infinite) samples, as $sq(\cdot)$ assigns them 0 probability.

We can evaluate any belief net B (or indeed, any candidate distribution) by its "(expected) L_2-accuracy", with respect to the actual query distribution $sq(\cdot)$ and underlying distribution P:

$$\mathrm{err}_{sq,\,P}(\,B\,) \quad = \quad \sum_{\mathbf{x},\mathbf{y}} sq(\,\mathbf{x}\,;\,\mathbf{y}\,)\,[P_B(\,\mathbf{x}\,|\,\mathbf{y}\,) - P(\,\mathbf{x}\,|\,\mathbf{y}\,)]^2 \qquad (1)$$

where the sum is over all assignments $\mathbf{x}, \mathbf{y}$ to all subsets $\mathbf{X}, \mathbf{Y}$ of variables, each $P(\,\mathbf{x}\,|\,\mathbf{y}\,)$ corresponds to the label for this query and $P_B(\,\mathbf{x}\,|\,\mathbf{y}\,)$ to the value that B assigns to this query. We will often write this simply as $\mathrm{err}(\,BN\,)$ when the distributions sq and p are clear from the context.

The learner's goal is to find a belief net that minimizes this score:

$$B^* \quad = \quad \underset{B}{\mathrm{argmin}}\{\ \mathrm{err}_{sq,\,p}(\,B\,)\ \}$$

While we typically do not have this $sq(\,\cdot\,)$ distribution, our learners usually will have a sample $LQ = \{\langle \mathbf{x}_i;\ \mathbf{y}_i;\ p_i\rangle\}_i$ from this "labeled query" distribution (Figure 1), which they can use to produce an approximation to this value for each net B:

$$\widehat{\mathrm{err}}^{(LQ)}(\,B\,) \ = \ \frac{1}{|LQ|} \sum_{\langle \mathbf{x};\,\mathbf{y};\,p\rangle \in LQ} [P_B(\,\mathbf{x}\,|\,\mathbf{y}\,) - p]^2 \quad (2)$$

and hence to identify a near-optimal $\widehat{B}$ whose error is, with high probability, close to the error of B^*.

Section 3 presents our results for learning in this ELQ ("explicitly labeled queries") model.

2.2.2 Unlabeled Queries/Forms + Samples

This ELQ model applies best if only a small number of queries will be asked (so we can explicitly enumerate the different queries and their labels), and the environment provides that relevant information. In some situations, however, there are too many possible queries, and the specific information is not available. As a realistic example, suppose a physician knows that 10% of his patients will complain of meningitis, and that in this situation, he will perform the k tests $\mathbf{U_{1..k}} = \{U_1, U_2, \ldots, U_k\}$. Given the results $\mathbf{r_{1..k}} = \langle r_1, \ldots, r_k\rangle \in \{t, f\}^k$ of these tests, he will then ask the "completely specified" query $P(\,\mathrm{meningitis}\,|\,\mathbf{U_{1..k}} = \mathbf{r_{1..k}}\,)$, and receive the label for this query, $\mathbf{V_{1..k}} \in [0,1]$.

In this situation, the physician may well know the (distribution over) these "query forms"; *e.g.*, 10% will be of the form $P(\,\mathrm{meningitis}\,|\,\mathbf{U_{1..k}}\,)$ (for some setting of the evidence); 23% will be $P(\,\mathrm{meningitis}\,|\,\mathbf{X_{1..\ell}}\,)$ based on some other tests $\mathbf{X_{1..\ell}}$; 5% will be $P(\,\mathrm{cancer}\,|\,\mathbf{W_{1..m}}\,)$; etc. However, he does not know:

Issue#1. the distribution over the particular $\mathbf{r_{1..k}}$ result values that he will observe. *E.g.*, of the 10% of the time associated with this $P(\,\mathrm{meningitis}\,|\,\mathbf{U_{1..k}}\,)$ form, how often will he see $\mathbf{U_{1..k}} = \langle t, t, \ldots, t, t\rangle$, as opposed to $\mathbf{U_{1..k}} = \langle t, t, \ldots, f\rangle$, and so forth, to $\mathbf{U_{1..k}} = \langle f, f, \ldots, f\rangle$?

Issue#2. the 2^k labels $\{\mathbf{V_{1..k}}\}$ for these 2^k queries.

Age	Gend	Smok	Temp	LivBio	Btest	EKG	Cancer	Menin
35	F	Y	⋆	⋆	⋆	⋆	Yes	⋆
25	M	N	⋆	⋆	⋆	⋆	No	⋆
⋆	⋆	⋆	⋆	True	⋆	⋆	Yes	⋆
⋆	F	⋆	⋆	False	True	⋆	⋆	Yes
⋆	F	⋆	High	True	⋆	⋆	⋆	No

Table 1: Sample of Implicitly Labeled Queries

Of course, we could try to force this within the ELQ-framework, by insisting that the learner also observe a big set of labeled "completely specified" queries, and use this sample to learn the required information about both "complete queries" and their labels. However, we would rather avoid the complication of getting such a query sample, especially as the domain expert may actually be able to explicitly supply the distribution over query forms.

As an alternative, we could address Issue#1 by assuming that this "sub-distribution" is uniform, or that it can be induced from the underlying distribution. Issue#2 is trickier, as here we want to avoid the need to explicitly specify an exponential amount of information, *viz.*, these 2^k different $\{V_{\mathbf{r}}\}$ values. Here, we could instead use a set of samples from the *underlying distribution*, then use this set of tuples to obtain estimates of the query labels.

Section 4 discusses the issues related to learning in this ULQS ("unlabeled queries plus samples") model.

2.2.3 Implicitly-Labeled Queries

Imagine again that an MD has first obtained the age and gender of his patient, together with the fact that she is a smoker, and then asked for the probability that this patient has cancer, given these symptoms — *i.e.*, posed the query "*What is* $P(\,\texttt{Cancer}\,|\,\texttt{Age} = 35,\ \texttt{Gender} = \texttt{F},\ \texttt{Smoke} = \texttt{T}\,)$*?*". As the value returned (0.65) is above a threshold, he will record this information in his log file, see (the first row of) Table 1. Notice (from the labels of the table's columns) that there are tests that were not performed, as well as other possible ailments not considered. As these values are not known, they are recorded as "⋆" in the first row.

The MD will later pose other queries, about this, or other patients; each will be recorded on subsequent rows of this emerging table. Hence, each row reflects a query about a single disease, based on the observations. If we know which of the attributes correspond to query variables (here the diseases Cancer and Meningitis), as opposed to the evidence nodes (Age, Gender, ..., EKG), we can then recover these queries, from this table. We can also get an idea of the label from that query, based on the assumption that this value was the argmax over the values of the query variables (here, $P(\,\texttt{cancer=Yes}\,|\,\texttt{female, 35yo, smoke}\,) > P(\,\texttt{cancer=No}\,|\,\texttt{female, 35yo, smoke}\,)$).

Note, in fact, that Table 1 corresponds to Figure 1, differing in that the table does not show the precise label of each query nor does it explicitly distinguish the query from the evidence variables.

Here, the value entered in the "Cancer" column is a deterministic function of the evidence: for each specific assignment to the evidence variables, there is a single value stored for the query variable. We also investigated an alternative "stochastic" model, where the MD found the true value of this query variable for each patient, and recorded that information. Hence, different patients, with the same body of evidence, could have different "cancer" values.

Notice this provides more information that in the deterministic model: if the MD sees n patients with the same symptoms (*e.g.*, 30 patients, all female 35-year-old smokers), and k of them had the same disease (*e.g.*, 20 had cancer), we could estimate the probability as k/n; *e.g.*,

$$\hat{P}(\,\texttt{Cancer}\,|\,\texttt{35yo, Fem, Smoker}\,) \;=\; 20/30$$

Under this condition, we could then use Section 2.2.1's "explicitly labeled query" model, based on the L_2 measure (Equation 1). In general, however, there are so many possible evidence variables and values that we anticipate getting (at most) one person with any set of symptoms. This means (our empirical estimates of) the labels would typically be 1/1. Moreover, we would only see the high probability assignments. We are therefore using a slightly different approach: Given any such set of these "implicitly labled queries" $IQ = \{\langle \mathbf{X}_i = \mathbf{x}_i, \mathbf{Y}_i = \mathbf{y}_i\rangle\}_i$ (Table 1), we define the "(empirical) conditional log likelihood" of a belief net B as

$$\widehat{\mathrm{CLL}}^{(IQ)}(\,B\,) \;=\; \frac{1}{|IQ|}\sum_{\langle\mathbf{x},\mathbf{y}\rangle\in IQ} \log(\,\hat{P}_B(\,\mathbf{x}\,|\,\mathbf{y}\,)\,) \qquad (3)$$

Of course, this is just an approximation to the "(true) conditional log likelihood"

$$\mathrm{CLL}_{sq}(\,B\,) \;=\; \sum_{\langle\mathbf{x},\mathbf{y}\rangle} sq(\,\mathbf{x}\,;\,\mathbf{y}\,) \;\times\; \log(\hat{P}_B(\,\mathbf{x}\,|\,\mathbf{y}\,)) \quad (4)$$

Our learner will then try to find the the belief net that maximizes this score. Section 5 investigates this ILQ ("implicitly labeled queries") learning framework.[2]

[2]This is, of course, an extremely simplistic model; see the discussion in Section 6. Also, [ZG99] relates the $\mathrm{err}(\cdot)$ and $\mathrm{CLL}.(\cdot)$ scores, and explains why we used the first for the ELQ and ULQS contexts, but the second for ILQ.

2.3 Contrast with other Learning Models

Much of the work in Machine Learning corresponds to *function approximation*: find a function that best matches a given set of input/output pairs. The canonical examples are learning classification or regression functions from a set of training data, using formalisms that range from neural nets to decision trees and rule sets [Mit97]. While our task does resemble this description — *cf.*, Equation 1 — our objective is different, in several ways: First, we allow different variables to serve as the classification variable for different instances. (*E.g.*, some queries may ask about the probability of cancer given some symptoms, while others ask about meningitis, and yet others, perhaps about someone's financial status, etc.) While we could attempt to learn k different classification functions, one for each query variable, this is not data-efficient, as it would be unable to exploit the rich structure of interrelations connecting these classification variables to each other, and to the various different symptom/evidence terms. (*E.g.*, the information in the labeled queries $LQ_{AB} = \{P(\,A\,) = p_1;\; P(\,A\,|\,B\,) = p_2;\; P(\,A\,|\,\neg B\,) = p_3;\; P(\,B\,) = p_B\}$ is interrelated, and should be exploited [Nil86].) We therefore want to learn a single structure that combines these different terms, which can therefore allow the information obtained for accommodating one class of queries to benefit the other classes.

Belief nets (Section 2.1) provide this capability. They also allow the eventual performance system to address, in a reasonable fashion, queries that were not present in the training sample. For example, a belief net that matches the first three labeled queries in LQ_{AB} will provide the appropriate response to the unseen "*What is $P(\,B\,)$?*" query; this similarly applies to unseen queries that involve subsets of the information present in the "training" queries. These BN-structures are also useful in other respects, as they provide a natural, and useful description of many situations; see the arguments in [Pea88], as well as the examples of deployed belief-net–based systems that appear in the UAI proceedings and elsewhere. We are therefore encoding the performance system as a belief net.

Of course, there are other BN-learning algorithms, that each attempt to produce a belief net from a set of data. These other algorithms, however, use a different type of training data — only from tuple data (but see the ILQ model, Section 5) — and have the different goal of finding the BN that is the best "match" to the training data; *e.g.*, whose likelihood (or posterior probability) is maximal [Hec95, Bun96], or which embodies all-and-only the conditional dependencies [GSSK87].

As discussed above, our goal is different: our learners seek the BN that produces the most *accurate* responses, over a distribution of queries. While this objective is weaker (as a BN that is a perfect model of the distribution will also have the most accurate possible responses), we argue that our model is often more appropriate. Of course, when the information in our training data (read "labeled queries") is not sufficient to completely specify the answer to the unseen queries, our learning algorithms, like all non-trivial learning algorithms, must generalize from the information presented. While the bias of our algorithm differs from that of most other BN-learning algorithms, it appears quite effective.

Our ILQ approach connects with [FGG97], which also seeks the BN that is best for some known distribution of queries, as it is trying to learn the optimal BN-based *classifier*. They also explain why the BN with maximal log-likelihood may not be the one with optimal performance on their specific task, and so evaluate BNs using a formula that is very similar to Equation 4's $\sum \log(P(\mathbf{x}\,|\,\mathbf{y}))$ expression (see also [BKRK97]). However, (1) we allow different variables to serve as the "classification variable" in different instances, while they consider only a single variable for all instances; (2) our sum depends on the query distribution; and (3) our actual algorithms explicitly try to optimize the conditional likelihood. Note also that they require each training tuple to be complete, while our model exploits the missing attribute values, as these omissions help define the actual query involved.

Finally, this paper extends our earlier [GGS97], which also claims the goal of a BN-learner should be a system that provides the appropriate answer to a set of queries. Here, we extend those earlier results by (1) providing additional theoretical results; (2) providing empirical validation, to both our earlier claims and the new ones; and (3) (also) considering slightly different models, including one based on the "conditional likelihood" score (Equation 4), which is more meaningful in many cases.

3 Learning in the ELQ Model

In [GGS97], we investigated this ELQ model in the PAC-framework [Val84], also focusing on the task of learning the CPtables for a given BN-structure. After proving that (typically) relatively few labeled-query tuples are required to provide the necessary information, that paper then proved it is NP-hard to find the values for the CPtables of a fixed BN-structure that have minimal error (Equation 2), wrt a given set of labeled queries.

3.1 *ELQ* Algorithm

[GGS97] therefore outlined a "hill-climbing" algorithm for this task, which we call "*ELQ*":

Let B be a belief net whose CPtable includes the value $e_{Q=q|\mathbf{R}=\mathbf{r}} = e_{q|\mathbf{r}} \in [0,1]$ as the value for the conditional probability of $Q = q$ given $\mathbf{R} = \mathbf{r}$. Let $[\mathbf{x}, \mathbf{y}] = [\mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y}]$ be a query, to which B assigns the probability $P_B(\mathbf{x}\,|\,\mathbf{y})$. Then the gradient of the empirical error function (Equation 2), for this single query, is

$$\begin{aligned} \frac{\partial\, \widehat{\mathrm{err}}^{(\langle \mathbf{x},\mathbf{y}\rangle)}(B)}{\partial\, e_{q|\mathbf{r}}} &= 2\,(P_B(\mathbf{x}\,|\,\mathbf{y}) - p) \times \\ & \frac{P_B(\mathbf{x}\,|\,\mathbf{y})}{e_{q|\mathbf{r}}}\,[P_B(q,\mathbf{r}\,|\,\mathbf{x},\mathbf{y}) - P_B(q,\mathbf{r}\,|\,\mathbf{y})] \end{aligned} \tag{5}$$

(Note the second line corresponds to $\frac{\partial P_B(\mathbf{x}\,|\,\mathbf{y})}{\partial e_{q|\mathbf{r}}}$.)

The *ELQ* algorithm adds up the contributions from all of the queries in our *LQ* sample, and then modifies the value of $e_{q|\mathbf{r}}$ by climbing some distance along this cumulative derivative, using straightforward and well-known techniques — see [BKRK97].

Optimizations: Unfortunately, evaluating the gradient requires computing conditional probabilities from a BN — a task known to be NP-hard to solve [Coo90], or even to approximate [Rot96, DL93]. We therefore looked for ways to avoid the need to compute these quantities. Equation 5 shows that we do not need to update $e_{q|\mathbf{r}}$ (at least, not due to the $[\mathbf{x}, \mathbf{y}]$ query) if the difference $P_B(\mathbf{x}\,|\,\mathbf{y}) - p$ is 0 (*i.e.*, if $P_B(\mathbf{x}\,|\,\mathbf{y})$ is correct) or if $P_B(q,\mathbf{r}\,|\,\mathbf{x},\mathbf{y}) - P_B(q,\mathbf{r}\,|\,\mathbf{y})$ is 0 (*i.e.*, if $\mathbf{y}$ "d-separates" $\mathbf{x}$ and $q, \mathbf{r}$). As our implementation includes these checks, it can avoid needless computations by updating only the other CPtables. In our experiments with the ALARM system (Section 3.2) for example, we found that only $\approx 10\%$ of the CPtable entries were relevant to any query, which meant we could avoid doing 90% of these expensive computations! Our implementation, based on JAVABAYES,[3] employs several other important optimizations, including the use of appropriate variable ordering, and caching of intermediate results; see [ZG99] for details.

Theoretical Claims: This process will climb to a CPtable whose empirical error is a local minimum. It is easy to show [GGS97] that, given a sufficiently large set of labeled query samples, this empirical error will be close to the true error, and hence *ELQ* will typically find good CPtable entries.

3.2 Experimental Results

We performed a set of experiments to validate the effectiveness of this *ELQ* algorithm.

[3] http://www.cs.cmu.edu/~javabayes/Home/

Example 1 (from [GGS97]) *As a simple "finger exercise", we considered the trivial belief net structure* $B_{AXC} = \boxed{A \to X \to C}$ *and specified that half of the labeled queries would be "*$P(\,C\,|\,A\,)$*, with label* 1*, and half would be* $P(\,C\,|\,\neg A\,)$ *with label* 0*. (That is,* A *should be equivalent to* C*.)*

Query	*Prob of asking*	*Label*
$P(\,C\,\|\,A\,)$	*1/2*	1.0
$P(\,C\,\|\,\neg A\,)$	*1/2*	0.0

(6)

We found that ELQ very quickly (typically in under 4 iterations) found the appropriate CPtables, with entries $e_{X|A} = e_{C|X} = 1.0$ *and* $e_{X|\bar{A}} = e_{C|\bar{X}} = 0.0$ *— i.e., making* $X \equiv A$ *and* $C \equiv X$.[4] *We will refer to the resulting BN as* B_{sq}*.* ∎

Example 2 (*ALARM *belief net B_{alarm} ***[BSCC89])*** *We first defined a realistic distribution over the queries: From [HC91], we know that a particular* 8 *of the variables typically appear as query variables, and a disjoint set of* 16 *variables appear as evidence. We therefore generated queries by uniformly selecting, as query, one of the* 8 *query variables, and then, for each of the* 16 *evidence variables, including it with probability* $1/4$ *— hence on average a query will include* $16/4$ *evidence variables. We then specify values for these evidence variables based on the natural joint distribution for these evidence variables. For example, suppose we have selected* Q *for the query variables, and* E_1, E_2 *as evidence variables. Then if* $P(\,E_1 = \texttt{t},\ E_3 = \texttt{f}\,) = 0.2$*, then the probability (given this form) of the query* $P(\,Q = ?\,|\,E_1 = \texttt{t},\ E_3 = \texttt{f}\,)$ *is* 0.2*. If* Q *has* k *values (e.g., boolean variables have the* $k = 2$ *values* $\{\texttt{t}, \texttt{f}\}$*), then we actually produce* k *copies of this specific query, e.g.,* $P(\,Q = \texttt{t}\,|\,E_1 = \texttt{t},\ E_3 = \texttt{f}\,)$ *and* $P(\,Q = \texttt{f}\,|\,E_1 = \texttt{t},\ E_3 = \texttt{f}\,)$*.*

We use this generator to produce a set of 500 *queries, each labeled with the true answer from* B_{alarm}*. We then evaluated this, using 10-fold cross-validation (i.e., 10 splits of* 450 *training elements and* 50 *testing); for each fold, we considered 3 different sets of initial CPtable entries — hence, we considered* 30 *runs. We then ran ELQ on each training set, to fill in the CPtables of the given* B_{alarm}*-structure. The "ELQ" line in Figure 2a shows how quickly this algorithm converged to near optimal* $err(\cdot)$ *values. To avoid further clutter, we did not plot the error-bars around the points; they were however extremely small — see [ZG99] for the details.*[5] ∎

[4]Of course, sometimes *ELQ* converged to the other optimum: $X \equiv \neg A$ and $C \equiv \neg X$.

[5]Except for the *APN* data, every point on every graph considers 30 runs, over various datasets.

Example 3 (*ASIA *belief net B_{asia} ***[LS88])*** *We also experimented with the* ASIA *dataset. Here, we generated queries completely at random; i.e., each of the* 8 *nodes was the query with probability 1/8, then each of the other nodes was selected as an evidence node with probability 1/2 (so there were on average 7/2 evidence nodes); they were assigned values according to the underlying probability. Figure 2b shows the converge rates, again averaged over* 30 *runs.* ∎

4 Learning in the ULQS Model

The ULQS learner has access to a set of *unlabeled queries UQ*, perhaps induced (see below) from a set of query-forms (which may be given explicitly, or alternatively induced from a set of observations, of either queries or, of query forms). It also has a sample of complete tuples S, drawn the underlying distribution P. The learner then tries to find the most accurate set of CPtable values, for a given BN-structure, based on the "expected L_2-accuracy" evaluation criterion (Equation 1).

One obvious approach is to reduce this task to the *ELQ* model, presented above. As noted above, we need to consider two issues: If we begin with a distribution over query forms, then the first challenge is finding the "subdistribution": for each query form, find the probability of asking each associated "completed query", which specifies all of the values of the evidence variables. As this will typically not be given (and even if such complete queries were given, it is still cumbersome to learn from them) we would need to make some assumptions. Two obvious candidate subdistributions are *uniform* (i.e., all possible assignments are equally likely) or *induced by underlying distribution*, i.e., $sq(\,\mathbf{X}=\mathbf{x}\,;\,\mathbf{Y}=\mathbf{y}\,) \;=\; \alpha\, P(\,\mathbf{Y}=\mathbf{y}\,)$ for some value of $\alpha = \alpha(\,\mathbf{X}, \mathbf{x}, \mathbf{Y}\,) \in \Re^+$. Note this model is meaningful if, for example, the MD asks everyone a certain set of questions.

Unfortunately, both approaches are problematic: Imagine the only query form was $P(\texttt{Pregnant}\,|\,\texttt{Gender})$ and imagine that 50% of patients are male. Both of the scheme mentioned above would therefore assume that the "probability of asking $P(\texttt{Pregnant}\,|\,\texttt{Gender} = \texttt{male})$" would equal the "probability of asking $P(\texttt{Pregnant}\,|\,\texttt{Gender} = \texttt{female})$". Of course, no MD's queries would match this pattern.

Fortunately, however, this "subdistribution" issue may be irrelevant. This comes from examining the second issue: finding the labels for the (unlabeled) queries in UQ. Here, we can use the sample S to estimate the required label for each "*What is* $P(\,\mathbf{X}=\mathbf{x}\,|\,\mathbf{Y}=\mathbf{y}\,)$*?*"

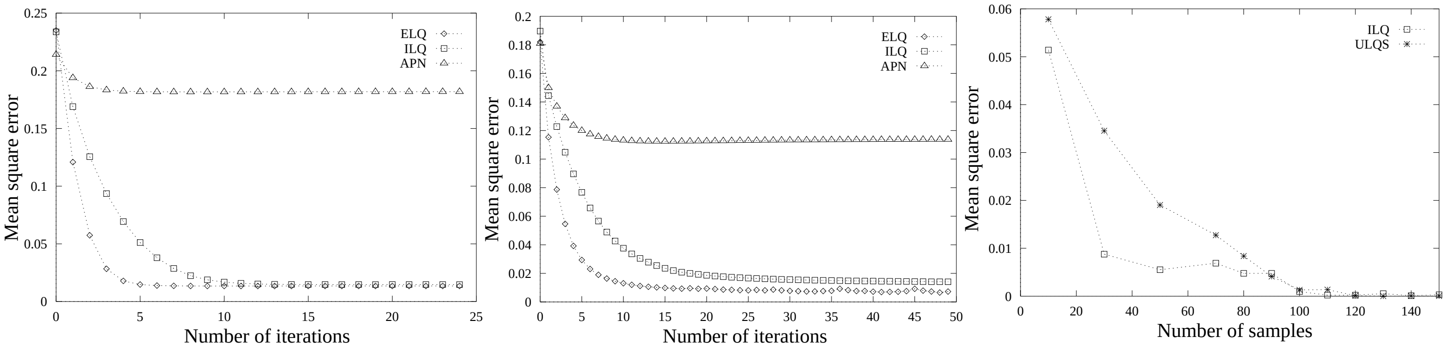

Figure 2: (a) Convergence Rates wrt B_{alarm}; (b) Convergence Rates, wrt B_{asia}; (c) Sample Complexity, wrt B_{AXC}

query, as the ratio of the number of examples in S that match both $\mathbf{X} = \mathbf{x}$ and $\mathbf{Y} = \mathbf{y}$, divided by the number that match $\mathbf{Y} = \mathbf{y}$. (And perhaps using a Laplacian correction to avoid dividing by 0.) We could then use our *ELQ* algorithm, using these $\hat{P}^{(S)}(\,\mathbf{x}_i \,|\, \mathbf{y}_i\,)$ estimates as the needed labels.

However, there may be a more efficient approach: *If the given BN-structure is correct* — that is, is an accurate I-map of the distribution — then we avoid using *ELQ*, and instead use the trivial "*OFE*" algorithm, which simply fills each CPtable with the "observed frequency estimates". This is, of course, the same algorithm that [CH92] proved would optimize the likelihood of the net, for a given structure, wrt a given set of training data. Notice this *OFE* algorithm is independent of the queries, and so it does not matter how we convert a distribution over query-forms to a distribution over completed queries.

To understand why *OFE* would be sufficient, just observe that the net obtained using these frequency estimates corresponds to the training data, and so would produce just those labels for the queries observed.

Of course, this news seems too good to be true, as it suggests we can replace the expensive iterative *ELQ* algorithm, which can only produce an approximate answer to the (in general NP-hard) task using a trivial algorithm, that requires only a single traversal of the training data, and which is guaranteed to produce the optimal solution. There are, however, two caveats: **First, *OFE* is only correct *if* the net structure is correct.** To illustrate this. . .

Example 1, con't: *Consider again asking the Equation 6 queries to the B_{AXC} network, but now imagine this structure was serious wrong, in that the intervening X is completely independent of A and C — i.e., $P(\,X \,|\, A\,) = P(\,X \,|\, \neg A\,) = P(\,C \,|\, X\,) = P(\,C \,|\, \neg X\,) = 0.5$.*

In this situation, the BN that most faithfully follows the underlying distribution, B_P, would have CPtable entries $e_{X|A} = e_{X|\bar{A}} = e_{C|X} = e_{C|\bar{X}} = 0.5$, with a performance score of err$(\,BN_P\,) = 0.25$. Now consider B_{sq}. While B_{sq} clearly has the X-dependencies completely wrong, its score is perfect — err$(\,B_{sq}\,) = 0.0$. ∎

To confirm that our *ELQ*-based approach would work here, we implemented a simple *ULQS* learner:

1. Use the sample of the underlying distribution to produce estimates of the query labels, then
2. use *ELQ* to find the best CPtable entries.

The *ULQS*-line in Figure 2c shows how quickly this algorithm learned, as a function of the number of tuple samples observed.

Note this "only correct structure caveat" explains why the general *ELQ* (and ULQS) task is NP-hard: Otherwise, there would be a trivial polynomial time algorithm (*OFE*) for computing an optimal CPtable for any fixed net structure — which cannot be (unless $P = NP$)! Of course, the *OFE*'s simplicity comes from the constraint that it must match every component of the distribution (*i.e.*, each CPtable entry). When the structure is wrong, however, a smarter *ULQS* algorithm can ignore this constraint for the irrelevant parts of the underlying distribution — in essense, it can use them as additional degrees of freedom, to use to better match the *queries*. As shown above, this is quite useful when the structure is wrong.

Of course, in this situation, a better approach would be to use the samples to produce a better belief net structure — one better matched to the data — and then using *OFE* (or actually, the *ULQS'* variant presented below) to fill in the CPtable entries.

The second problem with using *OFE* here is that *OFE* can be very (sample) inefficient.

Example 4 (from [GGS97]) *Assume the "reverse" naive-bayes structure B_{rnb}, shown in Figure 4,*

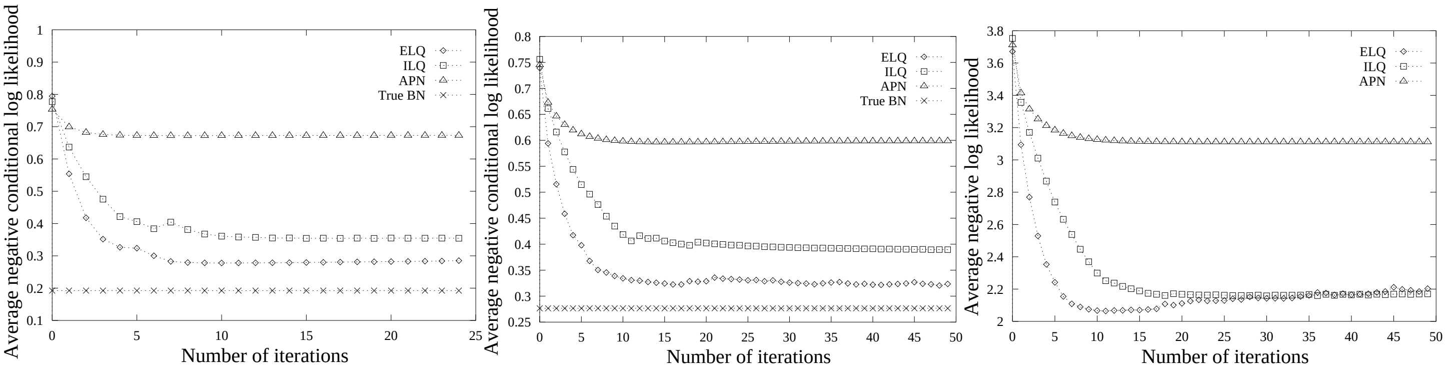

Figure 3: (a) CLL score for ALARM; (b) CLL Score for ASIA; (c) Log Likelihood for ASIA

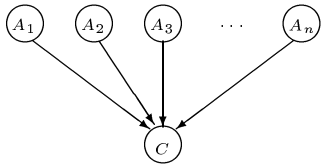

Figure 4: Belief net structure for Example 4

is a correct I-map. We therefore know OFE will produce the correct CPtable entries here, eventually.

It may however be very inefficient, for a given class of queries. For example, suppose we are only interested in the single query "$P(C = 1 \mid \{\}) = ?$". The simple OFE learner will not produce a good BN, capable of returning a good answer to this query, until it has obtained good estimates for (most of) C's 2^n CPtable entries. As each tuple-sample can contribute to the value of only a single such entry, this will require $> 2^n$ samples. (Note that this problem is not solved by using LaPlacian adjustments, as the accuracy of the resulting net B will produce an estimate $P_B(C = 1)$ that is poor unless $P(C = 1)$ happens to be near 0.5.) ∎

The general approach of using *OFE* here remains a good idea; the problem is the BN is too large. Here, it might make sense to first shrink the net, to a smaller structure that continues to "cover" all of the queries. Note that the (unlabeled) query forms — which may be given explicitly, and so be guaranteed to be exhaustive — are sometimes sufficient to determine this smallest size. This suggests the following *ULQS′* algorithm:

ULQS′(*UQ*: Unlabeled Query/QueryForms,
S: tuple-sample,
$G = \langle N, E\rangle$: BN-structure): BeliefNet

1. Find the smallest net-structure (perhaps a subset of G) that is sufficient to answer all of the query/query forms *UQ*
2. Use *OFE* to fill in the values of the CPtable entries.
3. Return resulting net (structure + CPtables)

For the B_{rnb} example, this algorithm would work wonderfully: The first step would reduce the structure to simply $\boxed{C}$, and the second would use the sample to fill in its (single) CPtable entry.

Unfortunately, this shrinking task is difficult, especially when we consider the details of the queries involved. That is,

Definition 1 *1. A belief net B "*entails*" a set of labeled queries $LQ = \{\langle q_i, v_i\rangle\}_i$ if $B(q_i) = v_i$ holds for all $\langle q_i, v_i\rangle \in LQ$.*

2. The size *of a belief net is the total number of parameters in all of its CPtables.*

3. Belief net B_1 is a subset *of belief net B_2 if B_1 and B_2 share the same nodes (variables) but B_1's arcs are a subset of B_2's arcs.* ∎

Unfortunately,[6]

Theorem 1 *Given a belief net B that entails a set of labeled queries LQ, it is NP-hard to find the smallest entailing subset of B.* ∎

Of course, one might argue the problem with the above approach is the constraint that forces the structure to be a substructure of the given network. However,

Theorem 2 *It is NP-hard to find the smallest belief net that is consistent with a given set of labeled queries LQ. This is true even if the labeled queries are known to be consistent (i.e., the labels were generated by some distribution), and when the conditioning part of each labeled query is an event of non-0 probability.* ∎

[6] All proofs appear in [ZG99].

5 Learning in the ILQ Model

Here, we consider the ILQ learner, which takes as input a belief net structure, and a set of "partial tuples" (Table 1), together with an annotation that identifies which variables are query variables, and which are evidence. It then tries to find the CPtable entries that produce the largest empirical conditional likelihood (Equation 3); given a sufficiently large set of implicitly-labeled query samples this will correspond to the BN with the best $\mathrm{CLL}_{sq}(\cdot)$ score.

Unfortunately,

Theorem 3 *It is NP-hard to find the values for the CPtables of a fixed BN-structure that produce the largest (empirical) conditional likelihood (Equation 3) for a given set of implicitly-labeled queries.* ∎

5.1 *ILQ* Algorithm

We therefore defined a simple hill-climbing algorithm, called *ILQ*, which uses the derivative of $\widehat{\mathrm{CLL}}^{(IQ)}(\,BN\,)$ wrt $e_{q|\mathbf{r}}$

$$\frac{\partial\, \widehat{\mathrm{CLL}}^{(IQ)}(\,BN\,)}{\partial\ e_{q|\mathbf{r}}} = \frac{1}{e_{q|\mathbf{r}}}\,[P_B(\,q,\mathbf{r}\,|\,\mathbf{x},\mathbf{y}\,) - P_B(\,q,\mathbf{r}\,|\,\mathbf{y}\,)] \qquad (7)$$

We then fitted this into a gradient ascent function, as with the *ELQ* algorithm. (Note again that this derivative is 0 if Q and $\mathbf{R}$ are d-separated from $\mathbf{X}$ and $\mathbf{Y}$.)

5.2 Empirical Exploration

We implemented this system, and explored its effectiveness in the various cases (all in the "stochastic" model"). After using this algorithm to instantiate a belief net structure, we evaluated the resulting BN B in two ways: either by computing its empirical $\widehat{\mathrm{err}}^{(B)}(\,S'\,)$ score on a hold-out dataset S', or by computing $\widehat{\mathrm{CLL}}^{(S')}(\,B\,)$ on that dataset.

Example 1, con't: *Consider again the B_{AXC} net and Equation 6 queries, in the context when the X variable is never present; hence the data looked like*

A	X	C
0	$\star$	0
1	$\star$	1
1	$\star$	1
0	$\star$	0
$\vdots$		

Suppose we knew that C was a query variable, and A was evidence. Our ILQ algorithm correctly instantiated the B_{AXC} network by making $A \equiv X \equiv C$ (i.e., producing B_{sq}), as desired. Figure 2c shows the dependency on the number of tuple-samples. ∎

There are many other algorithms for filling-in CPtables, from such "partial information". For example, the *APN* gradient ascent algorithm [BKRK97] computes the derivative of the "log likelihood" function, wrt each CPtable entry, then climbs along this gradient.

We therefore implemented *APN*[7], but found it did extremely poorly here, repeatedly returning $e_{x|a} = e_{x|\neg a}$ (*i.e.*, staying at the initial random assignment of these entries) and $e_{c|x} = e_{c|\neg x} = 0.5$. While this seems surprising, notice the log-likelihood derivative is 0 here. Unfortunately for *APN*, this is a saddle point and not a maximum. In essence, there is a huge basin collapsing to this manifold and associated local optimum, while the global optima at $\langle e_{x|a}, e_{x|\neg a}, e_{c|x}, e_{c|\neg x}\rangle = \langle 1,0,1,0\rangle$ or $\langle 0,1,0,1\rangle$ have extremely small basins.

We also found the exact same behavior for the obvious EM algorithm; see [ZG99].

Curiously, our algorithm, which is *not* trying to maximize likelihood, is doing better at maximizing likelihood than either the *APN* or EM, which are trying to maximize this score! The fault, of course, is not with the idea of maximizing likelihood, but instead with the shape and curvature of the space, and the ways these algorithms explore it.

Other Examples: We also experimented with the ALARM and ASIA datasets in this context. The results, when evaluated using the $\widehat{\mathrm{err}}^{(\cdot)}(\,\cdot\,)$ score, appear in Figures 2a,b labeled *ILQ*. Figure 3a,b show the associated $\widehat{\mathrm{CLL}}^{(\cdot)}(\,\cdot\,)$ scores.

All of the above figures also show *APN*'s performance, at these tasks.[8] Of course, this was unfair to *APN*, as its goal was to maximize likelihood $\sum_i \ln(x_i)$, where each x_i is the (partial) tuple present in the S samples — a goal that differs from our query-based error, and conditional likelihood, measures. Given our observations about the B_{AXC} situation, however, we decided to check how well these various algorithms did, evaluated using *APN*'s (log)likelihood measure. Figure 3c shows the results; notice here too our *ILQ* algorithm did significantly better that *APN*! We are currently investigating why this should be so.

[7] Our version differed slightly, by not using the conjugate gradient method.

[8] These results were based "simple" 10-fold CV, considering only one initial CPtable. We used a limited number of runs as we found this algorithm ran very slowly; this is partially because *APN* is forced to consider every CPtable; see Section 3.1.

6 Conclusions

Future Work: First, the above analyses deal with the task of filling in the CPtables of a given BN-structure. While this is an important subtask, a general learner should be able to use the available information (concerning both query and underlying distributions) to learn that structure as well.

Second, as noted above, the ILQ model is very naive; *e.g.*, each row of a realistic Table 1 will probably correspond to sequence of several queries, based on the successive tests run, until definitively confirming (or disconfirming) some disease. Each row may also reflect the MD's attempt to consider *several* diseases. Finally, the decision recorded (*e.g.*, "Cancer = Yes") should be based on utilities as well as probabilities.

Extending these observations, there may be ways to determine the query distribution from "first principles", based only on the underlying distribution and a cost model — perhaps by determining the decision-theoretic best queries in each situation, and assuming the diagnostician will ask these.

Finally, while we have presented three interrelated models for learning, there may well by others — that perhaps exploit both labelled queries *and* tuple sample. (For example, perhaps the learner should not "overfit" the learned BN to just the example queries it has seen; but should "extend" the BN, based on the tuple distribution.)

Which learning model is "correct", of course, depends critically on exactly what information is available to the learner, which is an empirical question. We recently posted a general call for additional real-world log-files and other sources that may reveal query distributions; we are eagerly awaiting responses.

Contributions: As noted repeatedly in Machine Learning and elsewhere, the goal of a learning algorithm should be to produce a "performance element" that will work well on its eventual performance task [SMCB77]. This paper considers the task of learning an effective belief net within this framework, and argues that the goal of a BN-learner should be to produce a BN whose "score" (L_2 or log-likelihood), over the distribution of queries, is optimal.

Our earlier paper [GGS97] proved that some parts of this task appear harder than the corresponding tasks, in the context of producing a BN that is optimal in more familiar "maximize likelihood" context. This paper advances our general understanding of these challenges, by considering other realistic learning models, and by demonstrating, empirically, that many of the associated algorithms do work well in practice — in some cases, actually work better than the algorithms developed for the other model. Such empirical evidence motivates us to continue exploring this rich, and potentially very useful, area.

References

[BKRK97] J. Binder, D. Koller, S. Russell, and K. Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29:213–244, 1997.

[BSCC89] I. Beinlich, H. Suermondt, R. Chavez, and G. Cooper. The ALARM monitoring system. In *Proceed, Second European Conference on Artificial Intelligence in Medicine*, 1989.

[Bun96] W. Buntine. A guide to the literature on learning probabilistic networks from data. *IEEE KDE*, 1996.

[CH92] G. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

[Coo90] G.F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2–3):393–405, 1990.

[DL93] P. Dagum and M. Luby. Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, 60:141–153, April 1993.

[FGG97] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29, 1997.

[GGS97] R. Greiner, A. Grove, and D. Schuurmans. Learning Bayesian nets that perform well. In *UAI-97*, 1997.

[GSSK87] C. Glymour, R. Scheines, P. Spirtes, and K. Kelly. *Discovering Causal Structure*. Academic Press, Inc., London, 1987.

[HC91] E. Herskovits and C. Cooper. Algorithms for Bayesian belief-network precomputation. In *Methods of Information in Medicine*, pages 362–370, 1991.

[Hec95] D. Heckerman. A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, 1995.

[LS88] S. Lauritzen and D. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *J. Royal Statistical Society*, 50:157–224, 1988.

[Mit97] T. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[Nil86] N. Nilsson. Probabilistic logic. *Artificial Intelligence*, 28(1):71–88, February 1986.

[Pea88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[Rot96] D. Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82(1–2), April 1996.

[SMCB77] R. Smith, T. Mitchell, R. Chestek, and B. Buchanan. A model for learning systems. In *IJCAI77*, pages 338–343, August 1977.

[Val84] Leslie G. Valiant. A theory of the learnable. *Comm. of ACM*, 27(11):1134–1142, 1984.

[ZG99] W. Zhou and R. Greiner. Learning accurate belief nets. Technical report, UofAlberta CS, 1999.