Active Learning for Structure in Bayesian Networks

Simon Tong
Computer Science Department
Stanford University
simon.tong@cs.stanford.edu

Daphne Koller
Computer Science Department
Stanford University
koller@cs.stanford.edu

Abstract

The task of causal structure discovery from empirical data is a fundamental problem in many areas. Experimental data is crucial for accomplishing this task. However, experiments are typically expensive, and must be selected with great care. This paper uses *active learning* to determine the experiments that are most informative towards uncovering the underlying structure. We formalize the causal learning task as that of learning the structure of a causal Bayesian network. We consider an active learner that is allowed to conduct experiments, where it *intervenes* in the domain by setting the values of certain variables. We provide a theoretical framework for the active learning problem, and an algorithm that actively chooses the experiments to perform based on the model learned so far. Experimental results show that active learning can substantially reduce the number of observations required to determine the structure of a domain.

1 Introduction

Determining the causal structure of a domain is frequently a key issue in many situations. *Bayesian networks (BNs)* [Pearl, 1988] are a compact graphical representation of joint probability distributions. They can also be viewed as providing a causal model of a domain [Pearl, 2000]. If we assume that the graphical structure of the BN represents the causal structure of the domain, we can formalize the problem of discovering the causal structure of the domain as the task of learning the BN structure from data.

Over the last few years, there has been substantial work on discovering BN structure from purely observational data. However, there are inherent limitations on our ability to discover the structure based on randomly sampled data. Experimental data, where we intervene in the model, is vital for a full determination of the causal structure. However, obtaining experimental data is often time consuming and costly. Thus the experiments must be chosen with care.

In this paper, we provide an *active learning* algorithm that selects experiments that are most informative towards revealing the causal structure. With active learning the choice of the next data case is based upon the results seen so far. The possibility of active learning can arise naturally in a variety of domains and in several variants. In *interventional* active learning, the learner can ask for experiments involving interventions to be performed. This type of active learning is the norm in scientific studies: we can ask for a rat to be fed one sort of food or another. An intervention in the model causes certain probabilistic dependencies in the model to be replaced by our intervention [Pearl, 2000] — the rat no longer eats what it would normally eat, but what we choose it to. By observing the results of this experiment, we can determine the direction of causal influence in cases where purely observational data is inadequate.

In such active learning settings, where we have the ability to actively select experiments, we need a mechanism that tells us which experiment to perform next. We present a formal framework for active learning in Bayesian networks, based on the principles of Bayesian learning. We maintain a distribution over Bayesian network structures, which is updated based on our data. We define a notion of *quality* of our distribution, and provide an algorithm that selects queries in a greedy way, designed to improve model quality as much as possible. We provide experimental results on a variety of domains, showing that our active learning algorithm can provide substantially more accurate estimates of the BN structure using the same amount of data. Interestingly, our active learning algorithm provides significant improvements even in cases where it cannot intervene in the model, but only select instances of certain types. Thus, it is applicable even to the problem of learning structure in a non-causal setting.

2 Learning Bayesian Networks

Let $\mathcal{X} = \{X_1, \ldots, X_n\}$ be a set of random variables, with each variable X_i taking values in some finite domain *Dom*$[X_i]$. A *Bayesian network (BN)* over $\mathcal{X}$ is a pair $(G, \boldsymbol{\theta}_G)$ that represents a distribution over the joint space of $\mathcal{X}$. G is a directed acyclic graph, whose nodes correspond to the random variables in $\mathcal{X}$ and whose structure encodes conditional independence properties about the joint distribution. We use $\mathbf{U}_i$ to denote the set of parents of X_i. $\boldsymbol{\theta}_G$ is a set of parameters which quantify the network by specifying the *conditional probability distributions (CPDs)* $P(X_i \mid \mathbf{U}_i)$.

The Bayesian network represents a joint distribution over the set of variables $\mathcal{X}$ via the *chain rule for Bayesian networks*: $P(X_1, \ldots, X_n) = \prod_i P(X_i \mid \mathbf{U}_i)$. Viewed as a probabilistic model, it can answer any query of the form $P(\mathbf{Y} \mid \mathbf{Z} = \mathbf{z})$ where $\mathbf{Y}$ and $\mathbf{Z}$ are sets of variables and $\mathbf{z}$ an assignment of values to $\mathbf{Z}$. However, a BN can also be viewed as a *causal model* [Pearl, 2000]. Under this perspective, the BN can also be used to answer *interventional queries*, which specify probabilities after we intervene in the model, forcibly setting one or more variables to take on particular values. In Pearl's framework, an intervention in a causal model that sets a single node $X := x$ replaces the standard causal mechanism of X with one where X is forced to take the value x. In graphical terms, this intervention corresponds to *mutilating* the model G by cutting the incoming edges to X. Intuitively, in the new model, X does not depend on its parents; whereas in the original model, the fact that $X = x$ would give us information (via evidential reasoning) about X's parents, in the experiment, the fact that $X = x$ tells us nothing about the values of X's parents. For example, in a fault diagnosis model for a car, if we observe that the car battery is not charged, we might conclude evidentially that the alternator belt is possibly defective, but if we deliberately drain the battery, then the fact that it is empty obviously gives us no information about the alternator belt. Thus, if we set $\mathbf{X} := \mathbf{x}$, the resulting model is a distribution where we mutilate G to eliminate the incoming edges to nodes in $\mathbf{X}$, and set the CPDs of these nodes so that $\mathbf{X} = \mathbf{x}$ with probability 1.

Our goal is to learn a BN structure from data. We assume that there are no hidden variables and in addition we make two further standard assumptions:

- **Causal Markov assumption:** The data is generated from an underlying Bayesian network $(G^*, \boldsymbol{\theta}^*)$ over $\mathcal{X}$.
- **Faithfulness assumption**: the distribution P^* over $\mathcal{X}$ induced by $(G^*, \boldsymbol{\theta}^*)$ satisfies no independences beyond those implied by the structure of G^*.

Our goal is to reconstruct G^* from the data. Clearly, given enough data, we can reconstruct P^*. However, in general, P^* does not uniquely determine G. For example, if our network G^* has the form $X \to Y$, then $Y \to X$ is equally consistent with P^*. Given only samples from P^*, the best we can hope for is to identify the *Markov equivalence class* [Pearl, 1988] of G: a set of network structures that induce precisely the same independence assumptions. In a Markov equivalence class, the skeleton of the network — the set connected (X, Y) pairs — is fixed; for some of the pairs, the direction of the edge is fixed, while the other edges can be directed either way [Spirtes *et al.*, 1993].

If we are given experimental as well as observational data, our ability to identify the structure is much larger [Cooper and Yoo, 1999]. Intuitively, assume we are trying to determine the direction of an edge between X and Y. If we are provided experimental data that intervenes at X, and we see that the distribution over Y does not change, while intervening at Y does change the distribution over X, we can conclude (based on the assumptions above) that the edge is $Y \to X$.

3 Bayesian learning with experimental data

As discussed in the introduction, our goal is to use active learning to learn the BN structure — learning from data where we are allowed to control certain variables by intervening at their values. We formalize this idea by assuming that some subset $\mathbf{Q}$ of the variables are *query variables*. The learner can select a particular instantiation $\mathbf{q}$ for $\mathbf{Q}$. The request $\mathbf{Q} := \mathbf{q}$ is called a *query*. The result of such a query is called the *response* and it is a randomly sampled instance $\mathbf{x}$ of all the *non-query* variables, conditioned on $\mathbf{Q} := \mathbf{q}$. In other words, $\mathbf{x}$ is the result of an experiment where we intervened in the model by setting $\mathbf{Q}$ to take the values $\mathbf{q}$; our assumptions then imply that $\mathbf{x}$ is sampled from the mutilated model described above.

We use a Bayesian framework to learn the BN structure. More precisely, we maintain a distribution over possible structures and their associated parameters. We begin with a prior over structures and parameters, and use Bayesian conditioning to update it as new data is obtained. Following [Heckerman *et al.*, 1995], we make several standard assumptions about the prior:

- **Structure Modularity**: The prior $P(G)$ can be written in the form: $P(G) = \prod_i P(Pa(X_i) = \mathbf{U}^G)$.
- **Parameter Independence**: $p(\boldsymbol{\theta}_G \mid G) = \prod_i p(\boldsymbol{\theta}_{X_i|\mathbf{U}_i} \mid G)$
- **Parameter Modularity**: For two graphs G and G', if $\mathbf{U}_i^G = \mathbf{U}_i^{G'}$ then: $p(\boldsymbol{\theta}_{X_i|\mathbf{U}_i^G} \mid G) = p(\boldsymbol{\theta}_{X_i|\mathbf{U}_i^{G'}} \mid G')$.

In this paper, we also assume that the CPD parameters are multinomials and that the associated parameter distributions are the conjugate Dirichlet distributions. However, our analysis holds for any distribution satisfying the parameter modularity assumption.

Given a complete, randomly sampled instance $\mathbf{d}$ over $\mathcal{X}$, using Bayes rule we have that the posterior distribution over G is proportional to: $P(\mathbf{d} \mid G)P(G)$. $P(\mathbf{d} \mid G)$ is the *marginal likelihood* of the data and can be expressed as an integral over all possible parameter values in G:

$$P(\mathbf{d} \mid G) = \int P(\mathbf{d} \mid G, \boldsymbol{\theta}_G) p(\boldsymbol{\theta}_G \mid G)\, d\boldsymbol{\theta}_G$$

Now, instead of having a complete random sample, suppose that we have an interventional query $\mathbf{Q} := \mathbf{q}$, and resulting response $\mathbf{x}$. We need to define how to update the distribution $P(G, \boldsymbol{\theta}_G)$ given this query and response. We break this into two problems by using the identity: $P(G, \boldsymbol{\theta}_G \mid \mathbf{Q} := \mathbf{q}, \mathbf{x}) = p(\boldsymbol{\theta}_G \mid \mathbf{Q} := \mathbf{q}, \mathbf{x}, G) \cdot P(G \mid \mathbf{Q} := \mathbf{q}, \mathbf{x})$. Thus we need to determine how to update the parameter density of a structure and also how to update the distribution over structures themselves.

For the first term in this expression, consider a particular network structure G and a prior distribution $p(\boldsymbol{\theta}_G)$ over the parameters of G. It is clear that we cannot use the resulting complete instance to update the parameters of the nodes $\mathbf{Q}$ themselves since we have forced $\mathbf{Q}$ to take on specific values. All other variables can, however, be updated with the complete instance. (Note that there is no issue of *selection bias* when updating the parameters of the ancestors of $\mathbf{Q}$ since all of incoming edges to query nodes $\mathbf{Q}$ are cut.) Thus, we define a variable Y to be *updateable in the context of an interventional query* $\mathbf{Q}$ if Y is not in $\mathbf{Q}$.

Our update rule for the parameter density is now very simple. Given a prior density $p(\boldsymbol{\theta}_G)$ and a response $\mathbf{x}$ from a query $\mathbf{Q} := \mathbf{q}$, we do standard Bayesian updating of the parameters, as in the case of randomly sampled instances, but we update only the Dirichlet distributions of updateable nodes. We use $p(\boldsymbol{\theta}_G \mid \mathbf{Q} := \mathbf{q}, \mathbf{x}, G)$ to denote the distribution $p'(\boldsymbol{\theta}_G)$ obtained from this algorithm; this can be read as "the density of $\boldsymbol{\theta}_G$ after performing query $\mathbf{q}$ and obtaining the complete response $\mathbf{x}$". Note that this is quite different from the density $p(\boldsymbol{\theta}_G \mid \mathbf{q}, \mathbf{x}, G)$ which denotes standard Bayesian conditioning. We also note that performing such an interventional update to the parameters still preserves parameter modularity.

Now consider the distribution over structures. We use $P(G \mid \mathbf{Q} := \mathbf{q}, \mathbf{x})$ to denote the posterior distribution over structures after performing the query and obtaining the response. The following theorem tells us how we can easily update the posterior over G given an interventional query:

Theorem 3.1 (Cooper and Yoo) *Given a query* $\mathbf{Q} := \mathbf{q}$ *and complete response* $\mathbf{x}$*, if* $P(G, \boldsymbol{\theta}_G)$ *satisfies* parameter independence *and* parameter modularity*, then:*

$$P(G \mid \mathbf{Q} := \mathbf{q}, \mathbf{x}) = \frac{P(G)}{P(\mathbf{x} \mid \mathbf{Q} := \mathbf{q})} \prod_{i : X_i \notin \mathbf{Q}} \mathit{Score}(X_i, \mathbf{U}_i^G \mid \mathbf{x}, \mathbf{q}),$$

with $\mathit{Score}(X_i, \mathbf{U} \mid \mathbf{d}) = \int P(x_i \mid \mathbf{u}, \boldsymbol{\theta}_{X_i|\mathbf{u}}) p(\boldsymbol{\theta}_{X_i|\mathbf{u}}) \, d\boldsymbol{\theta}_{X_i|\mathbf{u}}$.

Proof outline: Note that $\mathit{Score}(X_i, \mathbf{U} \mid \mathbf{d}) = P(x_i \mid \mathbf{u})$ where x_i and $\mathbf{u}$ are the values of X_i and $\mathbf{U}$ in the data instance $\mathbf{d}$. We have that $\prod_{i : X_i \notin \mathbf{Q}} \mathit{Score}(X_i, \mathbf{U}_i^G \mid \mathbf{x}, \mathbf{q}) = P(\mathbf{x} \mid \mathbf{Q} := \mathbf{q}, G)$. Finally, notice that $P(G) = P(G \mid \mathbf{Q} := \mathbf{q})$. □

4 Active Learning

Our goal in this paper is not merely to update the distribution based on interventional data. We want to *actively select* instances that will allow us to learn the structure better.

A *(myopic) active learner* ℓ is a function that selects a query $\mathbf{Q} := \mathbf{q}$ based upon its current distribution over G and $\boldsymbol{\theta}_G$. It takes the resulting response $\mathbf{x}$, and uses it to update its distribution over G and $\boldsymbol{\theta}_G$. It then repeats the process. We described the update process in the previous section. Our task now is to construct an algorithm for deciding on our next query given our current distribution P.

4.1 Loss function

As in the work of Tong and Koller [2001], a key step in our approach is the definition of a measure for the quality of our distribution over graphs and parameters. We can then use this measure to evaluate the extent to which various instances would improve the quality of our distribution, thereby providing us with an approach for selecting the next query to perform.

More formally, given a distribution over graphs and parameters $P(G, \boldsymbol{\theta}_G)$ we have a *loss function* $\mathit{Loss}(P)$ that measures the quality of our distribution over the graphs and parameters. Given a query $\mathbf{Q} := \mathbf{q}$ we define the *expected posterior loss* of the query as:

$$\begin{aligned} &\mathit{ExPLoss}(P(G, \boldsymbol{\theta}_G) \mid \mathbf{Q} := \mathbf{q}) \\ &\quad = \; E_{\mathbf{x} \sim P(\mathbf{X} \mid \mathbf{Q} := \mathbf{q})} \mathit{Loss}(P(G, \boldsymbol{\theta}_G \mid \mathbf{Q} := \mathbf{q}, \mathbf{x})). \quad (1) \end{aligned}$$

This definition immediately leads to the following simple algorithm: For each candidate query $\mathbf{Q} := \mathbf{q}$, we evaluate the expected posterior loss, and then select the query for which it is lowest. Note, however, that the expected loss appears to be computationally expensive to evaluate. We need to maintain a distribution over the set of structures, and the number of structures in $\mathcal{G}$ is super-exponential in the number of nodes. Furthermore, given a query, to compute the expected posterior loss we have to perform a computation over the set of structures for each of the exponential number of possible responses to the query.

To make this high-level framework concrete, we must pick a loss function. Recall that our goal is to learn the correct structure; hence, we are interested in the presence and direction of the edges in the graph. For two nodes X_i and X_j, there are three possible edge relationships between them: either $X_i \to X_j$, or $X_i \leftarrow X_j$ or $X_i \quad X_j$. Our distribution P over graphs and parameters induces a distribution over these three possible edge relationships. We can measure the extent to which we are sure about this relationship using the entropy of this induced distribution:

$$\begin{aligned} H(X_i \leftrightarrow X_j) = &-P(X_i \to X_j) \log P(X_i \to X_j) \\ &-P(X_i \leftarrow X_j) \log P(X_i \leftarrow X_j) \\ &-P(X_i \quad X_j) \log P(X_i \quad X_j) \qquad (2) \end{aligned}$$

The larger this entropy, the less sure we are about the relationship between X_i and X_j. This expression forms the basis for our *edge entropy* loss function:

$$\mathit{Loss}(P(G, \boldsymbol{\theta}_G)) = \sum_{i,j} H(X_i \leftrightarrow X_j) \qquad (3)$$

In certain domains we may be especially interest in determining the relationship between particular pairs of nodes. We can reflect this desire in our loss function by introducing scaling factors in front of different $H(X_i \leftrightarrow X_j)$ terms.

Now that we have defined the loss function for a distribution $P(G, \boldsymbol{\theta}_G)$, our task is to find an efficient algorithm for computing the expected posterior loss of a given query $\mathbf{Q} := \mathbf{q}$ relative to P. We note that P is our current distribution, conditioned on all the data obtained so far. Initially, it is the prior; as we get more data, we use Bayesian conditioning (as described above) to update P, and then apply the same algorithm to the posterior.

Our approach to obtaining a tractable algorithm is based on the ideas of Friedman and Koller [2000] — we first consider the simpler problem of restricting attention to network structures consistent with some total ordering, $\prec$. Then, we introduce a distribution over the orderings.

4.2 Analysis for a Fixed Ordering

Let $\prec$ be a total ordering of $\mathcal{X}$. We restrict attention to network structures that are consistent with $\prec$, i.e., if there is an edge $X \to Y$, then $X \prec Y$. Following [Friedman *et al.*, 1999], we also assume that each node X_i has a set $\mathbf{W}_i$ of at most m possible *candidate* parents that is fixed before each query round. In certain domains, we can use prior knowledge to construct $\mathbf{W}_i$; in others we can use a technique mentioned in [Friedman and Koller, 2000] where we can use the data itself to point out nodes that are more likely to be directly related to X_i. We define the set of candidate parents for a node X_i that are consistent with our ordering as: $\mathcal{U}_{i,\prec} = \{\mathbf{U} : \mathbf{U} \prec X_i, \mathbf{U} \subseteq \mathbf{W}_i\}$, where $\mathbf{U} \prec X_i$ if $Y \prec X_i$ for all $Y \in \mathbf{U}$. We note that the number of structures in induced by $\prec$ and $\mathbf{W}_i$ is still exponential in the number of variables in $\mathcal{X}$.

The key impact of the restriction to a fixed ordering is that the choice of parents for one node is independent of the choice of parents for another node [Buntine, 1991; Friedman and Koller, 2000]. Two important consequences are the following theorems [1], which give us closed form, efficiently computable expressions for key quantities:

Theorem 4.1 *Given a query $\mathbf{Q} := \mathbf{q}$, we can write the probability of a response $\mathbf{x}$ to our query as:*

$$\begin{aligned} & P(\mathbf{x} \mid \mathbf{Q} := \mathbf{q}, \prec) \\ = & \sum_{G \in \prec} \prod_i P(Pa(X_i) = \mathbf{U}_i^G) \prod_{j : X_j \notin \mathbf{Q}} Score(X_j, \mathbf{U}_j^G \mid \mathbf{x}, \mathbf{q}) \\ = & \lambda_{\mathbf{Q}} \prod_{i : X_i \notin \mathbf{Q}} \sum_{\mathbf{U} \in \mathcal{U}_{i,\prec}} P(Pa(X_i) = \mathbf{U}) Score(X_i, \mathbf{U} \mid \mathbf{x}, \mathbf{q}), \end{aligned}$$

where $\lambda_{\mathbf{Q}} = \prod_{i : X_i \in \mathbf{Q}} \sum_{\mathbf{U} \in \mathcal{U}_{i,\prec}} P(Pa(X_i) = \mathbf{U})$.

Theorem 4.2 *Given a query $\mathbf{Q} := \mathbf{q}$ and completion $\mathbf{x}$ we can write the probability of an edge $X_j \to X_i$ as:*

$$P(X_j \to X_i \mid \mathbf{Q} := \mathbf{q}, \mathbf{x}, \prec) = \frac{\sum_{\mathbf{U} \in \mathcal{U}_{i,\prec}, \mathbf{U} \ni X_j} P(Pa(X_i) = \mathbf{U}) Score(X_i, \mathbf{U} \mid \mathbf{x}, \mathbf{q})}{\sum_{\mathbf{U} \in \mathcal{U}_{i,\prec}} P(Pa(X_i) = \mathbf{U}) Score(X_i, \mathbf{U} \mid \mathbf{x}, \mathbf{q})},$$

where we define $Score(X_i, \mathbf{U} \mid \mathbf{x}, \mathbf{q}) = 1$ if $X_i \in \mathbf{Q}$.

Notice that since we are performing Bayesian averaging over multiple graphs the probability of an edge $X_i \to X_j$ will generally only be high if X_i is a *direct* cause of X_j rather than if X_i merely has some indirect causal influence on X_j.

Now, consider the expected posterior loss (Eq. (1)) given $\prec$:

$$\begin{aligned} & ExPLoss_{\prec}(P(G, \boldsymbol{\theta}_G) \mid \mathbf{Q} := \mathbf{q}) \\ = & E_{\mathbf{x} \sim P(\mathbf{X} \mid \mathbf{Q} := \mathbf{q}, \prec)} \sum_{i,j} H(X_i \leftrightarrow X_j \mid \mathbf{Q} := \mathbf{q}, \mathbf{x}, \prec). \end{aligned} \quad (4)$$

We can compute $H(X_i \leftrightarrow X_j \mid \mathbf{Q} := \mathbf{q}, \mathbf{x}, \prec)$ by using Theorem 4.2. Also, notice from Theorem 4.2 that the expression $H(X_i \leftrightarrow X_j \mid \mathbf{Q} := \mathbf{q}, \mathbf{x}, \prec)$ depends only on the values that $\mathbf{q}$ and $\mathbf{x}$ give to X_i, X_j, $\mathbf{W}_i$ and $\mathbf{W}_j$. Using this fact and then applying Theorem 4.1, we can rewrite the expected posterior loss as:

$$\begin{aligned} & ExPLoss_{\prec}(P(G, \boldsymbol{\theta}_G) \mid \mathbf{Q} := \mathbf{q}) \\ = & E_{\mathbf{x} \sim P(\mathbf{X} \mid \mathbf{Q} := \mathbf{q}, \prec)} \sum_{i,j} H(X_i \leftrightarrow X_j \mid x_i, x_j, \mathbf{w}_i, \mathbf{w}_j, \prec) \\ = & \sum_{i,j} \sum_{\mathbf{x}} P(\mathbf{x} \mid \mathbf{Q} := \mathbf{q}, \prec) H(X_i \leftrightarrow X_j \mid x_i, x_j, \mathbf{w}_i, \mathbf{w}_j, \prec) \\ = & \sum_{i,j} \sum_{\mathbf{x}} H(X_i \leftrightarrow X_j \mid x_i, x_j, \mathbf{w}_i, \mathbf{w}_j, \prec) \times \\ & \quad \lambda_{\mathbf{Q}} \prod_{k : X_k \notin \mathbf{Q}} \sum_{\mathbf{U} \in \mathcal{U}_{k,\prec}} P(Pa(X_k) = \mathbf{U}) Score(X_k, \mathbf{U} \mid \mathbf{x}, \mathbf{q}) \\ = & \lambda_{\mathbf{Q}} \sum_{i,j} \sum_{\mathbf{x}} \psi(x_i, x_j, \mathbf{w}_i, \mathbf{w}_j) \prod_{k : X_k \notin \mathbf{Q}} \phi(x_k, \mathbf{w}_k). \end{aligned} \quad (5)$$

Where,

$$\begin{aligned} \psi(x_i, x_j, \mathbf{w}_i, \mathbf{w}_j) &= H(X_i \leftrightarrow X_j \mid x_i, x_j, \mathbf{w}_i, \mathbf{w}_j, \prec) \\ \phi(x_k, \mathbf{w}_k) &= \sum_{\mathbf{U} \in \mathcal{U}_{k,\prec}} P(Pa(X_k) = \mathbf{U}) Score(X_k, \mathbf{U} \mid x_k, \mathbf{w}_k). \end{aligned}$$

This expression still involves summations over the exponential number of possible completions of a query. However, notice that for each i and j in Eq. (5), the summation over completions $\mathbf{x}$ resembles the expression for computing a marginal probability in Bayesian network inference where we are marginalizing out $\mathbf{x}$. In fact ψ and each ϕ can be regarded as factors and we can use standard graphical model inference procedures [Lauritzen and Spiegelhalter, 1988] to evaluate this expression effectively. The restriction to a candidate set of parents for each node ensures that each factor ϕ is over at most $(m + 1)$ variables, and each factor ψ over at most $2(m + 1)$ variables. After applying Bayesian network inference we end up with a factor over the variables $\mathbf{Q}$ where for each possible query $\mathbf{q}$ we have the value of the expression $\sum_{\mathbf{x}} \psi(x_i, x_j, \mathbf{w}_i, \mathbf{w}_j) \prod_{k : X_k \notin \mathbf{Q}} \phi(x_k, \mathbf{w}_k)$.

We need to perform such an inference for each i, j pair. However, since we restricted to at most m candidate parents, the number of possible edges is at most mn. Thus, the computational cost of computing the expected posterior loss for all possible queries is the cost of mn applications of Bayesian network inference.

4.3 Analysis for Unrestricted Orderings

In the previous section, we obtained a closed form expression for computing the expected posterior loss of a query for a given ordering. We now generalize this derivation by removing the restriction of a fixed ordering. The expression for the expected posterior loss can be rewritten as:

$$\begin{aligned} & ExPLoss(P(G, \boldsymbol{\theta}_G) \mid \mathbf{Q} := \mathbf{q}) \\ = & E_{\mathbf{x} \sim P(\mathbf{X} \mid \mathbf{Q} := \mathbf{q})} Loss(P(G, \boldsymbol{\theta}_G \mid \mathbf{Q} := \mathbf{q}, \mathbf{x})) \\ = & E_{\prec} E_{\mathbf{x} \sim P(\mathbf{X} \mid \mathbf{Q} := \mathbf{q}, \prec)} Loss(P(G, \boldsymbol{\theta}_G \mid \mathbf{Q} := \mathbf{q}, \mathbf{x})) \\ = & E_{\prec} E_{\mathbf{x} \sim P(\mathbf{X} \mid \mathbf{Q} := \mathbf{q}, \prec)} \sum_{i,j} H(X_i \leftrightarrow X_j \mid \mathbf{Q} := \mathbf{q}, \mathbf{x}). \end{aligned}$$

The expectation over orderings can be approximated by sampling possible orderings from our current distribution over graphs and parameters. As shown by Friedman and Koller [2000], sampling from orderings can be done very effectively using Markov chain Monte Carlo (MCMC) techniques.

The expression inside the expectation over orderings is very similar to the expected posterior loss of the query with a fixed ordering (Eq. (4)). The only difference is that we now must compute the entropy terms $H(X_i \leftrightarrow X_j \mid \mathbf{x}, \mathbf{Q} := \mathbf{q})$ without restricting ourselves to a single ordering. This entropy term is based on probability expressions for relationships between nodes:

$$\begin{aligned} & P(X_i \to X_j \mid \mathbf{Q} := \mathbf{q}, \mathbf{x}) \\ & \quad = E_{\prec \mid \mathbf{Q} := \mathbf{q}, \mathbf{x}} P(X_i \to X_j \mid \mathbf{Q} := \mathbf{q}, \mathbf{x}, \prec). \end{aligned} \quad (6)$$

Each of the terms inside the expectation can be computed using Theorem 4.2. Naively, we can compute the expectation for each query $\mathbf{Q} := \mathbf{q}$ and completion $\mathbf{x}$ by sampling orderings from $P(\prec \mid \mathbf{Q} := \mathbf{q}, \mathbf{x})$ and then computing $P(X_i \to X_j \mid \mathbf{Q} := \mathbf{q}, \mathbf{x}, \prec)$. Clearly, this approach

[1] Proofs are provided in the full version of this paper available at: robotics.stanford.edu/~stong/papers/tong_koller_ijcai01_full.ps

is impractical. However, we can use a simple approximation that substantially reduces the computational cost. Our general MCMC algorithm generates a set of orderings sampled from $P(\prec)$. In many cases, a single data instance will only have a small effect on the distribution over orderings; hence, we can often use our samples from $P(\prec)$ to be a reasonably good approximation to samples from the distribution $P(\prec \mid \mathbf{Q} := \mathbf{q}, \mathbf{x})$. Thus, we use our current set of sampled orderings to approximate Eq. (6). Note that this small approximation error will not accumulate since we are not using the approximation to update any of the parameters of our model, just merely to predict the value of possible queries in this current round.

We note that, as in the fixed ordering case, the entropy term $H(X_i \leftrightarrow X_j \mid \mathbf{Q} := \mathbf{q}, \mathbf{x})$ depends only on the values given to the variables $X_i, X_j, \mathbf{W}_i$ and $\mathbf{W}_j$. Thus, we can use the same Bayesian network inference method to compute the expression $E_{\mathbf{x} \sim P(\mathbf{X} \mid \mathbf{Q}:=\mathbf{q}, \prec)} \sum_{i,j} H(X_i \leftrightarrow X_j \mid \mathbf{Q} := \mathbf{q}, \mathbf{x})$.

4.4 Algorithm Summary and Properties

To summarize the algorithm, we first sample a set of orderings from the current distribution over graphs and parameters. We then use this set of orderings to compute the entropy terms $H(X_i \leftrightarrow X_j \mid \mathbf{Q} := \mathbf{q}, \mathbf{x})$. Next, for each ordering we compute $E_{\mathbf{x} \sim P(\mathbf{X} \mid \mathbf{Q}:=\mathbf{q}, \prec)} \sum_{i,j} H(X_i \leftrightarrow X_j \mid \mathbf{Q} := \mathbf{q}, \mathbf{x})$ using a standard Bayesian network inference algorithm to obtain a factor over all possible queries. We then average all of these query factors obtained from each ordering. The final result is a query factor that, for each possible query, gives the expected posterior loss of asking that query. We then choose to ask the query that gives the lowest expected posterior loss.

We now consider the computational complexity of the algorithm. For each ordering we need to compute $E_{\mathbf{x} \sim P(\mathbf{X} \mid \mathbf{Q}:=\mathbf{q}, \prec)} \sum_{i,j} H(X_i \leftrightarrow X_j \mid \mathbf{Q} := \mathbf{q}, \mathbf{x})$. This involves at most mn Bayesian network inferences. Each inference returns a factor over all possible queries and so the inference will take time exponential in the number of query variables. The time complexity of our algorithm to generate the next query is: $\mathcal{O}$(# of sampled orderings $\cdot\, mn \,\cdot$ cost of BN inference).

In addition, we need to generate the sampled orderings themselves. Friedman and Koller [2000] provide techniques that greatly reduce the cost of this process. They also show that the Markov chain mixes fairly rapidly, thereby reducing the number of steps in the chain required to generate a random sample. In our setting, we can reduce the number of steps required even further. Initially, we start with a uniform prior over orderings, from which it is easy to generate random orderings. Each of these is now the starting point for a Markov chain. As we do a single query and get a response, the new posterior distribution over orderings is likely to be very similar to the previous one. Hence, our old set of orderings is likely to be be fairly close to the new stationary distribution. Thus, a very small number of MCMC steps from each of the current orderings will give us a new set of orderings which is very close to being sampled from the new posterior.

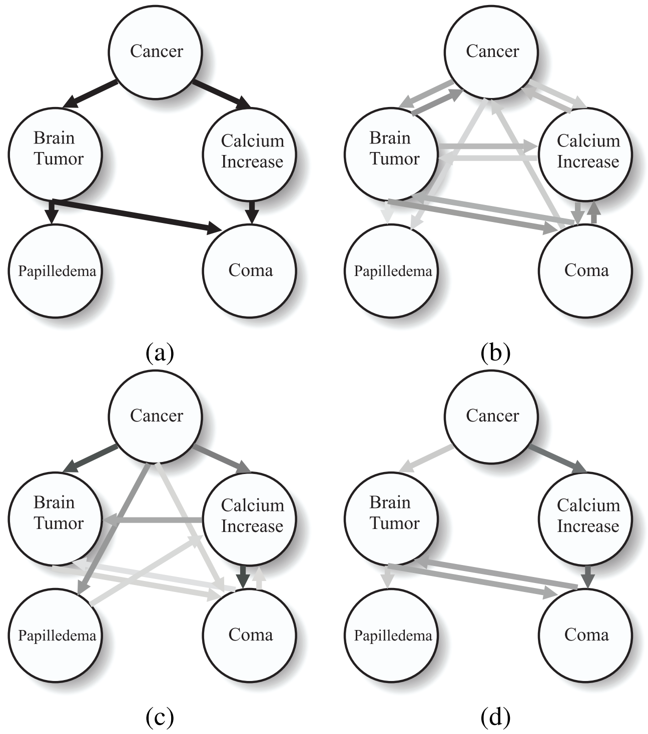

Figure 2: (a) Original **Cancer** network. (b) **Cancer** network after 70 observations. (c) **Cancer** network after 20 observations and 50 uniform experiments. (d) **Cancer** network after 20 observations and 50 active experiments. The darker the edges the higher the probability of edges existing. Edges with less than 15% probabilty are omitted to reduce clutter.

5 Experimental Results

We evaluated the ability of our algorithm to reconstruct the network structure by using data generated from a known network. We experimented with three commonly used networks: **Cancer**, with five nodes; **Asia**, with eight nodes; and **Car Troubleshooter**, with twelve nodes. For each test network, we maintained 50–75 orderings, as described above. We restricted the set of candidate parents to have size $m = 5$. It typically took a few minutes for the active method to generate the next query.

We compared our active learning method with both random sampling and uniform querying, where we choose a setting for the query nodes from a uniform distribution. Each method produces estimates for the probabilities of edges between each pair of variables in our domain. We compared each method's estimate with the true network G^* by using the L_1 *edge error* of the estimate:

$$\begin{aligned} \mathit{Error}(P) = \sum_{i,j>i} & I_{G^*}(X_i \rightarrow X_j)(1 - P(X_i \rightarrow X_j)) \\ & + I_{G^*}(X_i \leftarrow X_j)(1 - P(X_i \leftarrow X_j)) \\ & + I_{G^*}(X_i \;\; X_j)(1 - P(X_i \;\; X_j)) \end{aligned} \quad (7)$$

where $I_{G^*}(A) = 1$ if A holds in G^* and is zero otherwise.

We first considered whether the active method provides any benefit over random sampling other than the obvious additional power of having access to queries that intervene in the model. Thus, for the first set of experiments, we elimi-

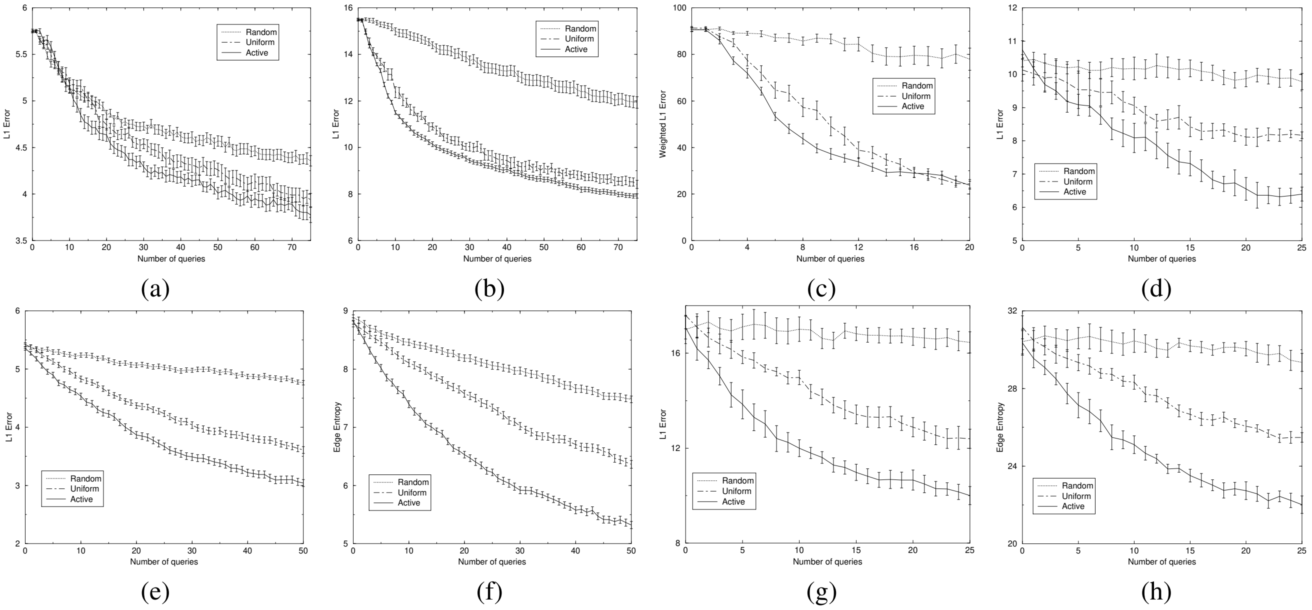

Figure 1: (a) **Cancer** with one query node. (b) **Car** with four query nodes. (c) **Car** with three query nodes and weighted edge importance. (d) **Asia** with any pairs or single or no nodes as queries. (e) **Cancer** with any pairs or single or no nodes as queries. (g) **Cancer** edge entropy. (f) **Car** with any pairs or single or no nodes as queries. (h) **Car** edge entropy. Legends reflect order in which curves appear. The axes are zoomed for resolution.

nated this advantage by restricting the active learning algorithm to query only roots of G^*. When the query is a root, a causal query is equivalent to simply selecting a data instance that matches the query (e.g., "Give me a 40-year-old male"); hence, there is no need for a causal intervention to create the response. Situations where we can only query root nodes arises in many domains; in medical domains, for example, we often have the ability to select subjects of a certain age, gender, or ethnicity, variables which are often assumed to be root nodes. All algorithms were informed that these nodes were roots by setting their candidate parent sets to be empty. In this batch of experiments, the candidate parents for the other nodes were selected at random, except that the node's true parents in the generating network were always in its candidate parent set.

Figures 1(a) and 1(b) show the learning curves for the **Cancer** and **Car** networks. We experimented with using uniform Dirichlet (BDe) priors and also more informed priors (simulated by sampling 20 data instances from the true network[2]). The type of prior made little qualitative difference in the comparative performance between the learning methods (the graphs shown are with uniform priors). In both graphs, we see that the active method performs significantly better than random sampling and uniform querying.

In some domains, determining the existence and direction of causal influence between two particular nodes may be of special importance. We experimented with this in the **Car** network. We modified the L1 edge error function Eq. (7) (and the edge entropy Eq. (3) used by the active method) to make determining the relationship between two particular nodes (the *FuelSubsystem* and *EngineStart* nodes) 100 times more important than a regular pair of nodes. We used three other nodes in the network as query nodes. The results are shown in Fig. 1(c). Again, the active learning method performs substantially better.

Note that, without true causal interventions, all methods have the same limited power to identify the model: asymptotically, they will identify the skeleton and the edges whose direction is forced in the Markov equivalence class (rather than identifying all edge directions in the true causal network). However, even in this setting, the active learning algorithm allows us to derive this information significantly faster.

Finally, we considered the ability of the active learning algorithm to exploit its ability to perform interventional queries. By using a simple extension to our analysis we permitted our active algorithm to choose to set any pair of nodes or any single node or no nodes at all. We compared this approach to random sampling and also uniformly choosing one of our possible queries (setting a single node, pair of nodes, or no nodes). Experiments were performed on the **Asia**, **Cancer**, and **Car** networks with an informed prior of 20 random observations. In this batch of experiments, we also experimented with different methods for choosing the candidate parents for a node X. As an alternative to using random nodes together with the true parents, we chose the $m = 5$ variables which had the highest individual mutual information with X. In practice this information could be obtained from observational data. Empirically, both methods of choosing the candidate parents gave very similar results, despite the fact that for one node in the **Car** network, a true parent of a node happened not to be chosen as a candidate parent with

[2]In general, information from observational data can easily be incorporated into our model simply by setting $\mathbf{Q}$ to be the empty set for each of the observational data instances. By Theorem 3.1, the update rule for these instances is equivalent to standard Bayesian updating of the model.

the mutual information method. We present the results using the mutual information criterion for choosing parents.

Figures 1(d), 1(e) and 1(g) show that in all networks our active method significantly outperforms the other methods. We also see, in Figures 1(f) and 1(h), that the prediction error graphs are very similar to the graphs of the edge entropy (Eq. (3)) based on our distribution over structures. This shows that the edge entropy is, indeed, a reasonable surrogate for predictive accuracy.

Figures 2(b), 2(c) and 2(d) show typical estimated causal edge probabilities in these experiments for random sampling, uniform querying and active querying respectively for the **Cancer** network (Fig. 2(a)). Figure 2(b) demonstrates that one requires more that just random observational data to learn the directions of many of the edges, and Fig. 2(d) shows that our active learning method creates better estimates of the causal interactions between variables than uniform querying. In fact, in some of the trials our active method recovered the edges and direction perfectly (when discarding low probablity edges) and was the only method that was able to do so given the limitation of just 50 queries. Also, our active method tends to be much better at not placing edges between variables that are only indirectly causally related; for instance in the network distribution learned by the active method (summarized in Fig. 2(d)), the probability of an edge from *Cancer* to *Coma* is only 3% and from *Cancer* to *Papilledema* only 4%.

6 Discussion and Conclusions

This paper introduces the problem of active learning of Bayesian network structure using interventional queries. We have presented a formal framework for this task, and a resulting algorithm for adaptively selecting which queries to perform. We have shown that our active method provides substantially better predictions regarding structure than both random sampling, and a process by which interventional queries are selected at random. Somewhat surprisingly, our algorithm achieves significant improvements over these other approaches even when it is restricted to querying roots in the network, and therefore cannot exploit the advantage of intervening in the model.

There is a significant body of work on the design of experiments in the field of optimal experimental design [Atkinson and Bailey, 2001]; there, the focus is not on learning the causal structure of a domain, and the experiment design is typically fixed in advanced, rather than selected actively. Active learning in Bayesian networks has been considered by Tong and Koller [2001] for parameter estimation where the structure is assumed to be known. There have been several studies on learning causal models from purely observational data [Spirtes *et al.*, 1993; Heckerman *et al.*, 1997]. Cooper and Yoo [1999] consider learning the structure of causal networks from a mixture of experimental and observational data, but in a non-active learning setting. Furthermore, although they derive a scoring metric for full networks, they only apply their technique to learn the relationship between single pairs of variables which they further assume are not confounded. On the other hand, our framework permits combining observational and experimental data for learning the structure over *all* variables in our domain, allowing us to distinguish the structure at a much finer level, taking into consideration both indirect causation and confounding influences.

There are many interesting directions in which we can extend our work, e.g., a treatment of continuous variables, temporal processes and different costs for different queries. Most interesting is the problem of dealing with hidden variables and missing data. We might use active learning to decide which extra variable to observe or which extra piece of missing data we should try to obtain in order to best learn the model. Another exciting direction is the potential of using active learning in order to try to uncover the existence of a hidden variable in our domain.

Acknowledgements The experiments were performed using the PHROG system, developed primarily by Lise Getoor, Uri Lerner, and Ben Taskar. The work was supported by DARPA's *Information Assurance* program under subcontract to SRI International, and by ONR MURI N00014-00-1-0637.

References

[Atkinson and Bailey, 2001] A. C. Atkinson and R. A. Bailey. One hundred years of the design of experiments on and off the pages of "Biometrika". *Biometrika*, 2001. In press.

[Buntine, 1991] W. Buntine. Theory refinement on Bayesian Networks. In *Proc. UAI*, 1991.

[Cooper and Yoo, 1999] G. F. Cooper and C. Yoo. Causal discovery from a mixture of experimental and observational data. In *Proc. UAI*, 1999.

[Friedman and Koller, 2000] N. Friedman and D. Koller. Being Bayesian about network structure. In *Proc. UAI*, 2000.

[Friedman *et al.*, 1999] N. Friedman, I. Nachman, and D. Pe'er. Learning Bayesian network structure from massive datasets: The "sparse candidate" algorithm. In *Proc. UAI*, 1999.

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

[Heckerman *et al.*, 1997] D. Heckerman, C. Meek, and G. Cooper. A Bayesian approach to causal discovery. Technical Report MSR-TR-97-05, Microsoft Research, 1997.

[Lauritzen and Spiegelhalter, 1988] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *J. Royal Statistical Society*, B 50(2), 1988.

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

[Pearl, 2000] J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2000.

[Spirtes *et al.*, 1993] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction and Search*. MIT Press, 1993.

[Tong and Koller, 2001] S. Tong and D. Koller. Active learning for parameter estimation in Bayesian networks. In *Proc. NIPS 13*, 2001. To appear.