Active Learning of Causal Bayes Net Structure

Kevin P. Murphy
Department of Computer Science
University of California
Berkeley, CA 94720-1776
murphyk@cs.berkeley.edu

Abstract

We propose a decision theoretic approach for deciding which interventions to perform so as to learn the causal structure of a model as quickly as possible. Without such interventions, it is impossible to distinguish between Markov equivalent models, even given infinite data. We perform online MCMC to estimate the posterior over graph structures, and use importance sampling to find the best action to perform at each step. We assume the data is discrete-valued and fully observed.

1 Introduction

There are essentially two kinds of approaches for learning the structure of Bayesian networks (BNs) from data. The first approach tries to find a graph which satisfies all the constraints implied by the empirical conditional independencies measured in the data [Pea00, SGS01]. The second approach searches through the space of graphs and uses some scoring metric to evaluate them [CH92, Hec95, Hec98], typically returning the highest scoring model found.

Both approaches to structure learning suffer from the fundamental problem that, even given infinite data, they can only identify the model up to Markov equivalence.[1] This is adequate for density estimation (i.e., if our goal is just to predict future observations), but is inadequate if our goal is causal discovery, since two BNs might be Markov equivalent and yet make different predictions about the consequences of interventions (e.g., $X \rightarrow Y$ and $Y \rightarrow X$ are Markov equivalent, but make very different assertions about the effect on Y of changing X).

The only way to distinguish members of the same Markov equivalence class is to perform experiments. By "experiments" we mean ideal interventions in the sense of Pearl [Pea00], i.e., the learning agent can clamp a subset of the variables to fixed values. For example, in the genetics domain, an experiment might consist of "knocking out" a gene, which we can think of as clamping it to a fixed value. The manipulation theorem [SGS01, Pea00] says that we can compute the consequences of such interventions by "cutting" all the arcs coming into the nodes which have been clamped by intervention, and then doing probabilistic inference in the "mutilated" graph in the usual way. (We are assuming all nodes are observed; if there are hidden nodes, we must use Pearl's "do calculus" to compute the consequences of interventions [Pea00]).

It is straightforward to modify existing Bayesian scoring methods to handle data obtained by interventional studies (in addition to the usual passive observational data). Specifically, we simply refrain from updating the parameters of the nodes that were clamped [CY99]. (The intuitive justification for this is that observing that a clamped node has a certain value does not tell us anything about how likely it is that that value would occur had we not forced it.) What has not been studied — with the notable exception of [TK01], which we discuss in Section 6 — is a way to decide which interventions to perform so as to learn the causal structure as quickly/ cheaply as possible. This is the goal of this paper.

We adopt a standard decision theoretic approach to the problem. The basic idea is to compute a posterior probability distribution over graph structures given some data, $P(G|D)$, and then, for each possible experimental action we can perform, compute the ex-

[1] Two graphs are Markov equivalent if they imply the same set of (conditional) independencies. For example, $X \rightarrow Y \rightarrow Z$, $X \leftarrow Y \rightarrow Z$ and $X \leftarrow Y \leftarrow Z$ are Markov equivalent, since they all represent $X \perp Z | Y$. In general, two graphs are Markov equivalent iff they have the same structure ignoring arc directions, and have the same v-structures [VP90]. (A v-structure consists of converging directed edges into the same node, such as $X \rightarrow Y \leftarrow Z$.)

pected utility of this action with respect to our current beliefs about the model. We then update our beliefs given the outcome of the experiment and repeat. The basic framework is discussed in Sections 2 and 3.

Since the number of directed acylic graphs (DAGs) grows super-exponentially with the number of nodes[2], we use a form of online MCMC to approximate our belief state, $P(G|D_{1:t})$, where $D_{1:t}$ is the data we have seen up to time-step t (i.e., t is the number of examples we have seen so far). In addition, computing the expected utility of an action requires enumerating all possible observations, which takes $O(2^n)$ time (assuming binary nodes). We approximate this using importance sampling. We discuss these approximations in Section 4. In Section 5, we show some experimental results using these approximations with three different Bayes nets.

Before diving into the technical details, we remark on the controversy concerning attempts to learn causality from data. Our claim is merely that statistical signatures in the data can suggest causal hypotheses that can be checked by experiment.

2 Active learning

The basic idea of our approach is illustrated in the influence diagram in Figure 1. We define the value (expected utility) of performing an action $A = a$ as

$$V(a) = \sum_{g \in \mathcal{G}} \sum_{y \in \mathcal{Y}} P(y|g,a,D)P(g|D)U(g,a,y,D)$$

where $U(\cdot)$ is a utility function, and $P(y|g,a,D)$ is the probability of generating observation y given that we have performed action a on graph g (e.g., if a is an intervention action, we need to mutilate g). $\mathcal{G}$ is the set of hypotheses (DAGs) we are considering, $\mathcal{Y}_{g,a}$ is the set of possible observations that g could generate after action a, and D is the past data. The best (myopic) action is defined to be $a^* = \arg\max_{a \in \mathcal{A}} V(a)$, where $\mathcal{A}$ is the set of possible actions (i.e., the set of values to which we can clamp the settable nodes).

If the goal is to infer the model structure, and all actions cost the same, we can use the following utility function: $U(g,a,y,D) = \log P(g|a,y,D)$. In Section 5, we will show experimentally that inferring the "correct" model structure is necessary if we are to predict the effect of interventions, but is *not* necessary if we are simply to predict future observations.

[2]No closed-form formula is known for the number of DAGs on n nodes, $f(n)$, but the first few values of f, for $n = 1, \ldots, 10$, are 1, 3, 25, 543, 29281, 3781503, 1.1×10^9, 7.8×10^{11}, 1.2×10^{15} and 4.2×10^{18} [Coo99]. A crude upper bound is $O(2^{n^2})$, the number of boolean matrices.

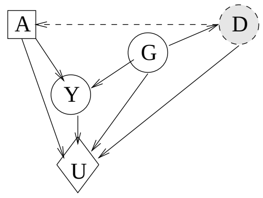

Figure 1: An influence diagram for one-shot experiment design. The dotted line is an informational arc, and specifies that the data D is observed before the action A (square node) is chosen. Y is a random variable that will be observed after action A. G is the unknown graph (its parameters are not shown, since they are integrated out). The diamond node U is the utility.

Before going into details, we will show that maximizing $\log P(g|a,y,D)$ is equivalent to maximizing the expected KL divergence between our posterior and our prior, or equivalently, to minimizing the conditional entropy of $H(G|Y,a,D)$ [Ber79], which are perhaps more familiar criteria.

$$V^* = \max_a E_Y KL(P(G|a,Y,D)||P(G|D))$$

$$= \max_a \sum_y P(y|a,D) \left[\sum_g P(g|a,y,D) \log \frac{P(g|a,y,D)}{P(g|D)} \right]$$

Since the denominator $P(g|D)$ is independent of a, we may drop it to get

$$V^* = \min_a - \sum_g P(g|D) \sum_y P(y|g,a,D) \log P(g|a,y,D)$$

3 Optimal algorithm

Let $\mathcal{G}$ be the (finite) set of all DAGs that we are considering. If all nodes are discrete (and hence $\mathcal{Y}$ and $\mathcal{A}$ are also finite sets), we can compute the optimal action by exhaustive enumeration, as follows.

for each $a \in \mathcal{A}$
    $V(a) = 0$
    for each $y \in \mathcal{Y}$
        Compute $P(y|g,a,D)\ \ \forall g \in \mathcal{G}$
        Compute $P(g|a,y,D)\ \ \forall g \in \mathcal{G}$
        $V(a) \mathrel{+}= \sum_g P(y|g,a,D)P(g|D) \log P(g|a,y,D)$

For completeness, we briefly summarise how to compute the various equations in the above algorithm. We use standard techniques for sequential Bayesian structure learning in fully observed, discrete domains (see [SL90, Bun94, Hec98] for details).

The expression $P(y|g,a,D)$ is the predictive density of g_a (the graph g modified by action a) given the past data D evaluated at y. This is given by the following equation:

$$\begin{aligned} & P(y|g,a,D) \\ = & \prod_{i=1}^{n}\prod_{j=1}^{q_i}\prod_{k=1}^{r_i}\int \theta_{ijk}^{1_{ijk}(y)} P(\theta_{ijk}|g,D)d\theta_{ijk} \\ = & \prod_{i=1}^{n}\prod_{j=1}^{q_i}\prod_{k=1}^{r_i}\overline{\theta}_{ijk}^{1_{ijk}(y)} \end{aligned}$$

where $1_{ijk}(y)$ is an indicator function that is 1 if the event $(X_i = k, \Pi_i = j)$ occurs in case y, and is 0 otherwise. r_i is the number of values X_i can take on, $q_i = \prod_{j \in \Pi_i} r_j$ is the number of values X_i's parents can take on. For unclamped nodes, we define $\theta_{ijk} = P(X_i = k|\Pi_i = j)$. For clamped nodes, we define $\theta_{ijk} = 1$ if X_i is clamped to k and $\theta_{ijk} = 0$ otherwise. (Hence $P(y|g,a,D)$ is 0 if y contradicts the values of the clamped nodes implied by a). If we assume the parameters of each node have independent Dirichlet priors with hyperparameters α_{ijk}, the posterior mean parameters are given by

$$\overline{\theta}_{ijk} = E[\theta_{ijk}|g,D] = \frac{\alpha_{ijk} + N_{ijk}}{\sum_{l=1}^{r_i} \alpha_{ijl} + N_{ijl}}$$

where $N_{ijk} = \sum_{y \in D} 1_{ijk}(y)$. In this paper, we use the priors $\alpha_{ijk} = 1/(r_i q_i)$; [HGC95] call this the BDeu metric. This ensures that Markov equivalent models have the same marginal likelihood given observational data alone, unlike using $\alpha_{ijk} = 1$.

Since many of the graphs in $\mathcal{G}$ will share the same families, we can use a cache to efficiently compute $P(y|g,a,D)$ for all $g \in \mathcal{G}$. (For example, if g_1 and g_2 only differ by a single edge addition/deletion, then $P(y|g_1,a,D)$ and $P(y|g_2,a,D)$ will only differ by one term.) Given the marginal likelihoods $P(y|g,a,D)$ and the priors $P(g|D)$ we can compute the posterior probability of each graph as follows:

$$P(g|a,y,D) = \frac{P(y|g,a,D)P(g|D)}{\sum_{g'} P(y|g',a,D)P(g'|D)}$$

In this paper, we use uniform structural priors, $P(g)$.

4 Sampling algorithm

The optimal algorithm assumes we can exhaustively evaluate $\mathcal{G}$, $\mathcal{Y}$ and $\mathcal{A}$. Typically this will be too expensive. We now discuss how to approximate each of these in turn.

4.1 Sampling graphs

The basic idea is to use the Metropolis-Hastings (MH) algorithm to draw samples from $P(G|D)$. The algorithm is summarized below, where $Q(G'|G)$ is the probability of proposing a move from G to G', B is the burn-in period, and N is the number of samples we want to draw. (See [MY95] for details.)

Choose G_1 somehow e.g., at random
For $t = 1, \ldots, B+N$
  Sample $G' \sim Q(\cdot|G_t)$
  Compute $R = \frac{P(G'|D)Q(G_t|G')}{P(G_t|D)Q(G'|G_t)}$
  Sample $u \sim \text{Unif}(0,1)$
  If $u < \min\{1, R\}$
    then $G_{t+1} := G'$
    else $G_{t+1} = G_t$
Return $G_{B+1}, \ldots, G_{B+N}$

The proposal distribution we use is to sample uniformly from the neighborhood of G, defined to be the set of all DAGs that differ from G by a single edge addition, deletion or reversal. (A way of quickly checking that the proposed graph is acyclic, based on the ancestor matrix, is described in [GC01].) In other words, $Q(G'|G) = 1/|nbd(G)|$, for $G' \in nbd(G)$, and $Q(G'|G) = 0$ for $G' \notin ndb(G)$, so

$$R = \frac{|nbd(G)|P(G')P(D|G')}{|nbd(G')|P(G)P(D|G))}$$

where the marginal likelihood is given by [CH92]

$$P(D|G) = \prod_{i=1}^{n}\prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + N_{ij})} \cdot \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + N_{ijk})}{\Gamma(\alpha_{ijk})}$$

The main advantage of this proposal distribution is that it is efficient to compute the Bayes factor $P(D|G')/P(D|G)$, since all but one (or two, in the case of an edge reversal) terms in the marginal likelihood ratio cancel.

[MAPV96] suggested searching the (smaller) space of (Markov) equivalence classes of DAGs, but this is inappropriate when we have interventional data. [FK00] suggested searching the (even smaller) space of total orderings of the nodes, marginalizing out the actual structure. Although this converges much faster, we chose not to pursue this technique, since we find it easier to design priors and heuristic proposal distributions in the space of graphs rather than orderings (see Section 6).

The MH algorithm as described above is appropriate for offline (batch) computation. However, we need to compute $P(G|D_{1:t})$ online (sequentially). We therefore combine the ideas of particle filtering (see e.g., [DdFG01]) with MH as follows. The belief state, $P(G|D_{1:t})$, is represented as a set of weighted particles (samples). When a new observation arrives, we apply a small number, B, of MH moves to each particle to get an approximation to $P(G|D_{1:t+1})$. The justification for this is that the new belief state is likely to be very similar to the old one (since we only have one new observation). We initialise by sampling from our prior $P(G)$.

Finally, we remark that in some domains, we have enough prior knowledge to construct a small set of hypothetical models, so we can feasibly enumerate all of $\mathcal{G}$ without needing to use MCMC.

4.2 Sampling observations

Even if we can find a small set of probable candidate models, the size of each model in this set, n, might be large, making it expensive to sum over all possible observations. (If all nodes are binary, there are $O(2^n)$ possible observations, and for continuous-valued nodes, there are an infinite number.) In this case, we can use importance sampling. The basic idea is to make the following approximation

$$E_Y f(Y) \approx \sum_{y \in \mathcal{Y}_s} w_y f(y)$$

where $f(y) = P(g|D) \log P(g|a, y, D)$, $\mathcal{Y}_s$ is a set of sampled y's drawn from some proposal distribution $Q(\cdot)$, and w_y are the normalized importance weights: $w_y \propto P(y|g, a, D)/Q(y)$.

Combining these two approximations, we modify the above active learning algorithm as follows.

Sample $\mathcal{G}_s \sim P(\cdot|D)$ using MCMC
for each $a \in \mathcal{A}$
    $V(a) = 0$
    Sample $\mathcal{Y}_s \sim Q(\cdot|a, D)$
    for each $y \in \mathcal{Y}_s$
        Compute $w_y(g) = P(y|g, a, D)/Q(y) \;\; \forall g \in \mathcal{G}_s$
        Compute $P(g|a, y, D) \;\; \forall g \in \mathcal{G}_s$
    Set $w_y(g) = \frac{w_y(g)}{\sum_{y'} w_{y'}(g)} \;\; \forall g \in \mathcal{G}_s$
    $V(a) {+}{=} \sum_g \sum_y w_y(g) P(g|D) \log P(g|a, y, D)$

The simplest proposal distribution for y is the uniform one: $Q(y) = 1/2^{n-k}$, where we have assumed that all nodes are binary and the action a has clamped k nodes. The optimal proposal is of course $Q(y) = P(y|g, a, D)$. In this case, the algorithm becomes

Sample $\mathcal{G}_s \sim P(\cdot|D)$ using MCMC
for each $a \in \mathcal{A}$
    $V(a) = 0$
    for each $g \in \mathcal{G}_s$
        Sample $\mathcal{Y}_s \sim P(\cdot|g, a, D)$
        Set $w_y = 1/|\mathcal{Y}_s|$
        for each $y \in \mathcal{Y}_s$
            Compute $P(g'|a, y, D) \;\; \forall g' \in \mathcal{G}_s$

$$V(a) {+}{=} w_y P(g|D) \log P(g|a, y, D)$$

Since this is $O(|\mathcal{G}_s|^2)$, it is typically too expensive. A cheaper approximation would be to sample from the mixture distribution $\sum_g P(y|g, a, D)P(g|D)$. In this paper, however, we just sample from the uniform distribution.

4.3 Sampling actions

Finally, we consider the $\max_a$ computation. If we can clamp up to k nodes simultaneously, and all nodes are binary, then $|\mathcal{A}| \leq \binom{n}{k} 2^k = O(n^k)$. Since it is often difficult or expensive to clamp many nodes simultaneously, k is often small; hence n^k is a low order polynomial. Nevertheless, since we must do $O(|\mathcal{G}_s| \cdot |\mathcal{Y}|_s)$ computations per action, this can be expensive. We are currently working on some heuristic methods to avoid this brute force enumeration.

For continuous-valued nodes, the number of actions is in principle infinite, although in practice it might be reasonable to discretize the action space. For example, in the genetics domain, we might be able to clamp a node to its "wildtype" (mean) value μ, to "overexpress" it (clamp it to $\mu + \sigma$, where σ is the standard or to "underexpress" it (clamp it to $\mu - \sigma$). (See [BMI99] for an interesting MCMC approach to selecting continuous-valued actions.)

4.4 How many samples?

The question of how many samples we need to take is an interesting one. The key insight is that, for action selection, it is the relative values of $V(a)$ that matter. This idea has been exploited in [OK00] to reduce the number of samples used (see also the "Hoeffding races" approach of [MM93]). However, in this paper, we just use a fixed number of samples.

5 Results[3]

We compare the behavior of the active learner with two other algorithms: passive observation, and random interventions. We assume we can clamp up to two nodes simultaneously to any of their legal values. There are various ways to compare the algorithms. Since the posterior over all graphs is too big to easily visualize, one way of compressing it, following [TK01], is to compute the L_1 edge error induced by belief state at time

[3] All of the experiments were performed in Matlab using the Bayes Net Toolbox, which is available from `www.cs.berkeley.edu/ ~murphyk/Bayes/bnt.html`.

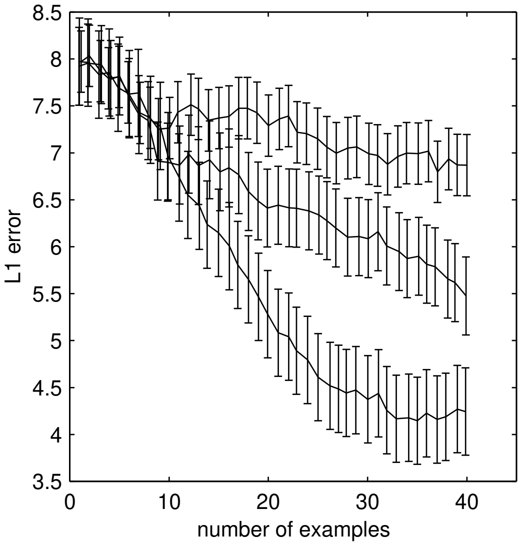

Figure 2: L_1 edge error against number of examples for the cancer network using random CPD parameters from a uniform distribution. The lines from top to bottom represent passive, random and active learning. We used 10 sampled observations, 100 sampled graphs. and averaged over 5 trials.

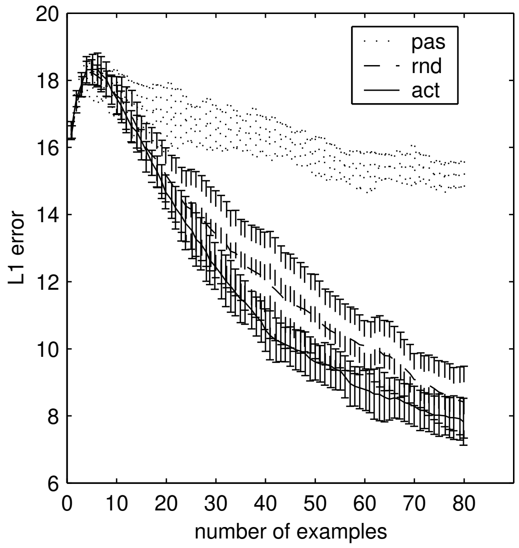

Figure 3: L_1 edge error against number of examples for the Asia network using the published CPD parameters. The lines from top to bottom represent passive, random and active learning. We computed $\sum_y$ exactly, used 100 sampled graphs, and averaged over 10 trials.

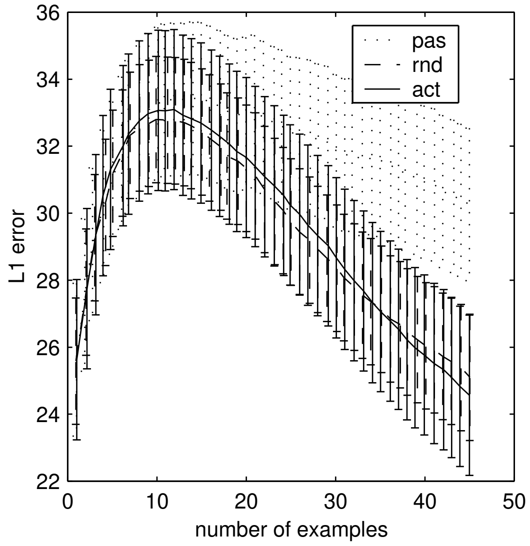

Figure 4: L_1 edge error against number of examples for the car troubleshooter network using random CPD parameters from a Beta(0.2,0.2) distribution. The top, dotted line is the passive learner, the other two lines are the random and active learners. We used 100 sampled observations, 300 sampled graphs, and averaged over 5 trials.

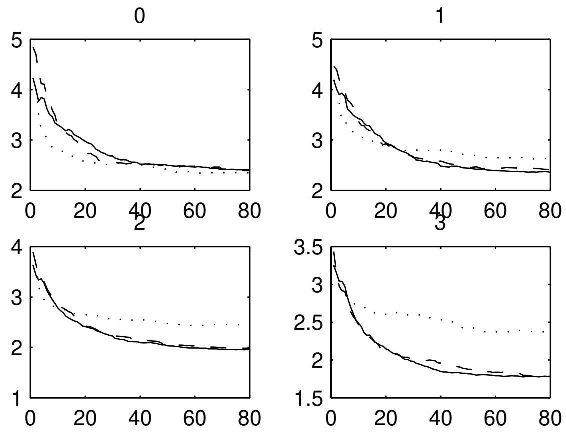

Figure 5: Prediction performance after 0, 1, 2 or 3 interventions to the Asia network. The vertical axis is negative log likelihood, the horizontal axis is number of examples. The dotted line is the passive learner, the other two lines are the random and active learners. We averaged over 10 trials.

t, $P_t = P(G|D_{1:t})$:

$$\begin{aligned}\text{err}_t^{L_1}(P_t) &= \sum_{i=1}^{n}\sum_{j=i+1}^{n} I_{G^*}(X_i \rightarrow X_j)(1 - P(X_i \rightarrow X_j)) \\ &+ I_{G^*}(X_i \leftarrow X_j)(1 - P(X_i \leftarrow X_j)) \\ &+ I_{G^*}(X_i \perp X_j)(1 - P(X_i \perp X_j))\end{aligned}$$

where $I_{G^*}(X_i \rightarrow X_j)$ is an indicator variable that is 1 if there is an edge from X_i to X_j in the true graph structure G^*, and $X_i \perp X_j$ means there is no connection between X_i and X_j. (Henceforth we shall call the true model the "oracle".)

Another metric is to evaluate the ability of the models to predict future observations, either drawn from the oracle, or generated in response to an intervention in the oracle. To do this, we simply generate a test set of size M using the (possibly mutilated) oracle, and compute its average negative log likelihood according to the current belief state of the learner:

$$\text{err}_t^{\text{pred}}(P_t) = \frac{-1}{M}\sum_{m=1}^{M}\sum_{g} \log P(y_m|g)P(g|D_{1:t})$$

We consider performing 0, 1 and 2 interventions in the oracle, which corresponds to clamping 0, 1 and 2 nodes simultaneously to random values, and then sampling the other nodes. A test set consists of $M = 20$ such cases.

To be comparable with [TK01], we test our algorithm on three commonly-used networks: the 5 node cancer network [FMR98], the 8 node Asia network [LS88], and the 12 node car trouble-shooter network [HBR94]. All networks have binary nodes with multinomial conditional probability distributions (CPDs). For the Asia network, we used the published parameters. For the other networks, we sampled the CPD parameters from a Dirichlet distribution. By changing the parameters of the Dirichlet, we can control the degree of determinism. Specifically, as $\alpha_{ijk} \rightarrow 0$, the CPTs become more deterministic; for $\alpha_{ijk} = 1$, the CPT entries are chosen uniformly on $[0, 1]$ (subject to the sum-to-one constraint); and as $\alpha_{ijk} \rightarrow \infty$, each row of the CPT tends towards the maximum entropy distribution of $[1/r_i, \ldots, 1/r_i]$.

In Figures 2, 3 and 4, we show how the L_1 error decreases as the number of examples increases. For the cancer and Asia networks, we see that the active learner identifies the structure much more quickly than the random or passive learners. For the car trouble shooter network, however, the active learner does not seem to do any better than the random learner; presumably this is because we are not sampling for long enough. This is something we plan to investigate further.

In Figure 5, we plot the prediction performance of the learners for the Asia network. In Figure 5(a) we see that the passive learner can predict observations from the unmutilated oracle slightly better than the other learners (initially, at least), even though it has the "wrong" structure. However, Figures 5(b-d) indicate that knowing the right structure helps to predict the effect of interventions, as expected, especially as the number of clamped nodes increases. We have found this to be true for the other models as well, especially in the more deterministic parameter regime.

6 Related work

Most previous work on active learning has been in the context of classification, regression or function optimization. In the case of linear regression, the minimum posterior entropy criterion discussed in Section 2 is called D-optimality, and can be maximized in closed form. Many other results (such as A-optimility, G-optimality, etc.) have been derived for the linear regression model, in both a Bayesian and non-Bayesian setting (see e.g., [CV95] for a review). There has also been work on active learning for non-linear regression models (e.g., neural networks [Mac92] and locally weighted regression [CGJ96]), where the objective is to minimize the expected variance of the predictor. In the above works, the active learner can choose any point in the input space. By contrast, in the query filtering paradigm, the learner can choose to see the label of certain items from a stream of inputs (see e.g., [FSST97]).

In the PAC setting, [Ang88] showed how the ability to ask questions reduces the problem of identifying certain kinds of boolean functions from NP-complete to polynomial time. [BHH95] and [TR98] have exended this to active learning of tree-structured boolean functions, where the internal nodes are hidden. [AKMM98] have some results concerning upper and lower bounds on the number of experiments necessary to learn (possibly cyclic) boolean networks. [ITK00] discusses active learning techniques for learning boolean networks using an entropy-based cost function.

The most closely related work is that of Tong and Koller [TK01], who also use a decision theoretic framework for BN structure learning. (We shall henceforth refer to this as the "TK" algorithm.) The TK algorithm builds upon [FK00], who do MCMC over total orderings of the nodes, instead of over DAGs. The space of orderings is "only" of size $n!$, much smaller than the space of DAGs, which is a space of size $O(2^{n^2})$. The key insight is that, conditioned on an ordering $\prec$, the parents for each node can be chosen independently (because there is no longer any global acyclicity constraint), and hence can marginalized out. TK extend this by showing how, conditioned on $\prec$, one can perform $\sum_y$ (needed for the expected utility computation) using the variable elimination algorithm.

The advantages of our algorithm over TK are its simplicity, the fact that it makes no assumptions about the form of the loss function, and the fact that it can incorporate prior structural knowledge. The loss function used by TK is $\text{Loss}(P(G|a, y, D)) = \sum_{ij} H_{ij}$, where H_{ij} is the entropy associated with the edge distribution between nodes i and j. Although this is fairly intuitive, it is not as theoretically well motivated as $H(G|Y, a, D)$, which we used above.

The advantage of the TK algorithm over ours is speed. Firstly, it computes $\sum_y$ analytically, instead of using importance sampling. And secondly, it uses MCMC in the smaller space of orderings. However, it is not clear how big the former speedup is: the cost of variable elimination depends on the induced width of the graph, which in turn depends on the summation ordering and the structure of the graphs induced by the set of all possible families; hence the induced width can become quite large. Also, while [FK00] have shown that MCMC in the space of orderings converges faster than in the space of DAGs, we believe that for either approach to be really practical, one will need strong prior knowledge, and it is easier to specify a prior as $P(G)$ rather than the much less natural $P(G|\prec)$.

7 Future work

There are many issues we would like to pursue in the future, the main ones being: continuous variables, missing data, online learning, and dynamical systems. and applications to biology.

For continuous variables, we plan to use linear-Gaussian CPDs, possibly with non-linear basis functions, as discussed above. (The use of such non-linearities makes this different from the global jointly Gaussian approach of [HG95].) For missing data, we plan to use sampling (data augmentation). The actions might now also consist of choosing to measure a hidden variable, as in classical value-of-information computations. For online learning, we can no longer keep the whole dataset D, nor can we store the (expected) sufficient statistics for all possible models. One approach would be to keep the statistics just for a "fringe" of probable models, as in [FG97].

It is straightforward to adapt the above techniques to learn the structure of a dynamic Bayesian network (DBN) from time series data c.f., [FMR98]. If we only allow arcs between time-slices ("diachronic" arcs), the parents for each node can be chosen independently, as in feature subset selection. Hence we can get the advantages of the TK algorithm without the need to sample node orderings.

Finally, we eventually hope to apply the algorithm to the problem of experiment design for inferring the structure of gene regulatory networks.

Acknowledgements

I would like to thank Stuart Russell and Michael Jordan for comments on an earlier draft of this paper.

References

[AKMM98] T. Akutsu, S. Kuhara, O. Muruyama, and S. Miyano. Indentification of gene regulatory networks by strategic gene disruptions and gene overexpressions. In *SIAM Symposium on Discrete Algorithms*, 1998.

[Ang88] D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.

[Ber79] J. Bernardo. Expected information as expected utility. *Annals of Statistics*, 7:686–690, 1979.

[BHH95] N. Bshouty, T. Hancock, and L. Hellerstein. Learning Boolean read-once formulas over generalized bases. *J. Comp. and Systems Sciences*, 50(3):521–542, 1995.

[BMI99] C. Bielza, P. Mueller, and D. Ríos Insua. Decision analysis by augmented probability simulation. *Management Science*, 45(7):995–1007, 1999.

[Bun94] W. L. Buntine. Operations for learning with graphical models. *J. of AI Research*, pages 159–225, 1994.

[CGJ96] D. Cohn, Z. Ghahramani, and M. Jordan. Active learning with statistical models. *J. of AI Research*, 4:129–145, 1996.

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

[Coo99] G. Cooper. An overview of the representation and discovery of causal relationships using Bayesian networks. In C. Glymour and G. Cooper, editors, *Computation, Causation & Discovery*. MIT Press, 1999.

[CV95] K. Chaloner and I. Verdinelli. Bayesian experimental design: A review. Technical report, Univ. Minnesota, 1995. www.stat.umn.edu/PAPERS/tr607.html.

[CY99] G. Cooper and C. Yoo. Causal discovery from a mixture of experimental and observational data. In *UAI*, 1999.

[DdFG01] A. Doucet, N. de Freitas, and N. J. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer Verlag, 2001.

[FG97] N. Friedman and M. Goldszmidt. Sequential update of Bayesian network structure. In *UAI*, 1997.

[FK00] N. Friedman and D. Koller. Being Bayesian about network structure. In *UAI*, 2000.

[FMR98] N. Friedman, K. Murphy, and S. Russell. Learning the structure of dynamic probabilistic networks. In *UAI*, 1998.

[FSST97] Y. Freund, H. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28:133–168, 1997.

[GC01] P. Giudici and R. Castelo. Improving Markov chain Monte Carlo model search for data mining. *Machine Learning*, 2001. To appear.

[HBR94] D. Heckerman, J. Breese, and K. Rommelse. Troubleshooting under uncertainty. Technical Report MSR-TR-94-07, Microsoft Research, 1994.

[Hec95] D. Heckerman. A Bayesian approach to learning causal networks. In *UAI*, 1995.

[Hec98] D. Heckerman. A tutorial on learning with Bayesian networks. In M. Jordan, editor, *Learning in Graphical Models*. MIT Press, 1998.

[HG95] D. Heckerman and D. Geiger. Learning Bayesian networks: a unification for discrete and Gaussian domains. In *UAI*, volume 11, pages 274–284, 1995.

[HGC95] D. Heckerman, D. Geiger, and M. Chickering. Learning Bayesian networks: the combination of knowledge and statistical data. *Machine Learning*, 1995.

[ITK00] T. Ideker, V. Thorsson, and R. Karp. Discovery of regulatory interactions through perturbation: inference and experimental design. In *Proc. of the Pacific Symp. on Biocomputing*, 2000.

[LS88] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their applicatins to expert systems. *J. R. Stat. Soc. B*, B(50):127–224, 1988.

[Mac92] D. MacKay. Information-based objective functions for active data selection. *Neural Computation*, 4:589–603, 1992.

[MAPV96] D. Madigan, S. Anderson, M. Perlman, and C. Volinsky. Bayesian model averaging and model selection for Markov equivalence classes of acyclic graphs. *Communications in Statistics: Theory and Methods*, 25:2493–2519, 1996.

[MM93] O. Maron and A. W. Moore. Hoeffding races: Accelerating model selection search for classification and function approximation. In *NIPS-6*, 1993.

[MY95] D. Madigan and J. York. Bayesian graphical models for discrete data. *Intl. Statistical Review*, 63:215–232, 1995.

[OK00] L. Ortiz and L. Kaelbling. Sampling methods for action selection in influence diagrams. In *AAAI*, 2000.

[Pea00] J. Pearl. *Causality: Models, Reasoning and Inference*. Cambridge Univ. Press, 2000.

[SGS01] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. MIT Press, 2001. 2nd edition.

[SL90] D. J. Spiegelhalter and S. L. Lauritzen. Sequential updating of conditional probabilities on directed graphical structures. *Networks*, 20, 1990.

[TK01] S. Tong and D. Koller. Active learning for structure in Bayesian networks. In *IJCAI*, 2001. Submitted.

[TR98] P. Tadepalli and S. Russell. Learning from examples and membership queries with structured determinations. *Machine Learning*, 32:245–295, 1998.

[VP90] T. Verma and J. Pearl. Equivalence and synthesis of causal models. In *UAI*, 1990.