BIOINFORMATICS

Vol. 20 no. 17 2004, pages 2934–2942
doi:10.1093/bioinformatics/bth337

Reconstruction of gene networks using Bayesian learning and manipulation experiments

*Iosifina Pournara and Lorenz Wernisch**

Department of Crystallography, Birkbeck College, University of London, Malet Street, London, WC1E 7HX, UK

Received on December 4, 2003; revised on April 14, 2004; accepted on May 14, 2004
Advance Access publication June 4, 2004

ABSTRACT

Motivation: The analysis of high-throughput experimental data, for example from microarray experiments, is currently seen as a promising way of finding regulatory relationships between genes. Bayesian networks have been suggested for learning gene regulatory networks from observational data. Not all causal relationships can be inferred from correlation data alone. Often several equivalent but different directed graphs explain the data equally well. Intervention experiments where genes are manipulated can help to narrow down the range of possible networks.

Results: We describe an active learning algorithm that suggests an optimized sequence of intervention experiments. Simulation experiments show that our selection scheme is better than an unguided choice of interventions in learning the correct network and compares favorably in running time and results with methods based on value of information calculations.

Availability: Algorithms are available from the authors on request.

Contact: l.wernisch@bbk.ac.uk

INTRODUCTION

Knowledge of mRNA levels under different conditions can help in understanding how the expression levels of each gene depend on the external stimuli and on the expression levels of other genes. With high throughput experimental methods, such as DNA microarrays, mRNA expression levels of a number of genes can be measured simultaneously. Recently, various variants of probabilistic networks have been suggested for extracting information on genetic regulation from a set of gene expression profiles (Segal *et al*., 2001; Hartemink *et al*., 2001). Bayesian networks are a sparse representation of joint probability distributions on a set of variables (gene expression levels in our case). Some methods for learning networks are based on partial correlations (Pearl, 2000; Spirtes *et al*., 2001), while others, such as Bayesian learning, are based on scoring schemes (Cooper and Herskovits, 1992; Heckerman *et al*., 1995; Friedman and Koller, 2000). The resulting networks are used to infer causal relationships between genes and to predict the outcome of specific experiments.

Most approaches focus on learning Bayesian networks from *observational data*, i.e. data collected without any biological or experimental interference on the level of individual genes. One difficulty with such data is that it is only possible to identify an equivalence classes of networks. An equivalence class is a set of different networks implying the same set of conditional independencies and the same probability score. On the other hand, Cooper and Yoo (1999) have shown that by including data from interventions one can distinguish between such networks. We use the term *experimental data* to indicate that one or more genes have been manipulated, i.e fixed to a particular value.

Manipulation experiments can be costly and time consuming. Active learning algorithms identify nodes which are most informative about the causal relationships in the underlying network. Ideker *et al*. (2000) suggested an algorithm that identifies informative nodes in Boolean networks. Different Boolean networks can generate the same variable assignment and thus suffer from a similar problem as Bayesian networks.

Other approaches deal with Bayesian networks on discrete domains. The algorithm of Tong and Koller (2001) selects interventions that reduce the entropy of the probability of alternative edge orientations. Once a total ordering of the nodes is sampled, the expected posterior loss in entropy due to interventions can be calculated efficiently. A similar though more general approach is discussed in Murphy (2001) where the expected information gain of an action is calculated. In both approaches the expectation calculations are based on observational data, data obtained in previous experiments, and data predicted to result from the action in question. Both require considerable sampling for each possible action.

In this paper, we present a new method for choosing a sequence of experiments. Our algorithm evaluates expected costs more directly than the above approaches obviating the computationally expensive predictive sampling step, which becomes quickly prohibitive for larger networks. Essentially, our algorithm considers the structure of equivalence classes and selects manipulations which tend to split large and highly

*To whom correspondence should be addressed.

likely equivalence classes into many small ones. Despite its conceptual simplicity the algorithm achieves results comparable to or even better than that obtained by previous methods. Moreover, we suggest extensions of active learning algorithms to continuous variables.

In the following, we provide some background information on learning Bayesian networks. We then describe our algorithm. For the evaluation of our algorithm we need to resort to simulations, which allow us to compare the outcome of the algorithm with alternative variable selection schemes. Finally, we describe an efficient enumeration and a sampling algorithm for equivalence classes.

BAYESIAN NETWORKS

Given a set of variables $U = \{x_1, \ldots, x_n\}$, a Bayesian network is a graphical representation of a joint probability distribution over all states of U. It consists of a directed acyclic graph (DAG) G whose nodes are random variables and whose edges (arcs) correspond to direct probabilistic dependencies between nodes (Fig. 1). Bayesian networks are given a causal interpretation by assuming that direct dependencies are actually causal relationships. The intended interpretation here is that variables represent gene expression levels, and that arcs between variables represent regulatory relationships.

Each variable follows a probability distribution that depends on its parents only; parameters characterizing these distributions are denoted by Θ. The graph G encodes the Markov assumption which states that a variable x_i is independent of its non-descendants given its parents π_{x_i}. The joint probability distribution can be decomposed as

$$p(x_1, \ldots, x_n \mid \xi) = \prod_{i=1}^{n} p(x_i \mid \pi_{x_i}, \xi)$$

where ξ denotes some background knowledge.

Data are given in the form of a set $D = \{C_1, \ldots, C_m\}$ where C_l comprises values for all the variables in U for the l-th observation. A scoring metric calculates the posterior probability of G in given data D

$$p(G \mid D, \xi) = p(D \mid G, \xi) p(G \mid \xi) / p(D \mid \xi)$$

where the marginal likelihood is

$$p(D \mid G, \xi) = \int p(D \mid \Theta, G, \xi)\, p(\Theta \mid G, \xi)\, d\Theta$$

$p(G \mid \xi)$ is the prior probability of the network and $p(\Theta \mid G, \xi)$ are the parameter priors. The reader interested in the details is referred to Heckerman and Geiger (1995).

Bayesian learning consists of two ingredients: a probabilistic scoring function indicating how well a network explains the data, and a procedure for searching for a network or a set of networks with high scores. A popular scoring metric is the Bayesian likelihood equivalent metric (Be) which computes

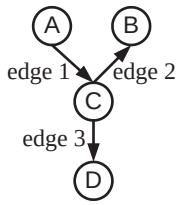

Fig. 1. An example of a Bayesian network consisting of four variables.

the posterior probability of a network given either discrete data (BDe metric) or continuous data (BGe metric) (Cooper and Herskovits, 1992; Heckerman *et al*., 1995; Geiger and Heckerman, 1994).

Each pair of variables x, y can have three different causal relationships: x causes y ($x \rightarrow y$), y causes x ($x \leftarrow y$), or neither x nor y causes the other ($x \perp y$). The number of possible networks for a set of variables can grow exponentially with the number of nodes making exhaustive searches difficult [a recursive equation for counting acyclic graphs is given, e.g. in Melancon *et al*. (2000)]. In this paper we use a simple search heuristic—greedy hill-climbing—which adds, deletes or reverses edges which result in a maximum increase in the score. Other methods include simulated annealing, bootstrap and Markov chain Monte Carlo sampling (see Friedman *et al*., 1999; Friedman and Koller, 2000).

Transition sequence equivalent networks

To score discrete networks not only with respect to observational data but also with respect to experimental data, Cooper and Yoo (1999) and later Tian and Pearl (2001) suggested a modification of the BDe metric, the BDeES metric (ES stands for experimental sequence). In this metric all parent–child configurations for a manipulated child are ignored. They no longer reflect possible dependencies, which are severed by holding the manipulated variable fixed to a certain value. Cooper and Yoo (1999) described the metric for experimental data for discrete domains. We extend this metric to Gaussian networks in Appendix.

Two Bayesian networks may represent exactly the same set of dependencies and conditional independencies. For example, the networks $x \rightarrow y \rightarrow z, x \leftarrow y \leftarrow z$, and $x \leftarrow y \rightarrow z$ imply the same conditional independence assertion that x is independent of z given y. Such DAGs or networks are called *equivalent*. In the following criterion for equivalent networks a *v-structure* is a triplet (x, y, z) with converging arcs $x \rightarrow y \leftarrow z$ while x and z are not connected.

Equivalent networks (Verma and Pearl, 1990). Two network structures are equivalent if and only if they have the same structure ignoring arc directions and the same v-structures.

A representation of an equivalence class is given by a partially directed acyclic graph (PDAG) consisting of directed and undirected edges. Only edges with the same direction in

all networks of the class (compelled edges in the terminology of Chickering, 1995) are directed in the PDAG. Edges with two different orientations in the equivalence class are undirected (reversible edges). We use an algorithm by Chickering (1995) to generate the PDAG of a DAG.

One cannot distinguish between equivalent networks based on a database D of observations alone (e.g. Heckerman *et al*., 1995). However, experimental data from interventions allow us to distinguish some of them. For example, the networks $x \rightarrow y \rightarrow z$ and $x \leftarrow y \rightarrow z$ are equivalent. Assuming the first network is correct, a change of y to a specific value will often cause a change in z but not in x. Thus, the probabilistic score obtained when data from a manipulation of y are added will be higher for $x \rightarrow y \rightarrow z$ than for $x \leftarrow y \rightarrow z$. However, even with manipulation some networks are still indistinguishable.

Transition sequence equivalent networks (Tian and Pearl, 2001). Two networks G_1 and G_2 are transition sequence equivalent (TS-equivalent) if and only if they have the same structure ignoring arc directions, the same set of v-structures, and the same sets of parents for all manipulated variables.

If we assume that z is manipulated instead of y in the previous example, then—since z is the effect of y in both networks—neither x nor y will be affected in either network. That is, both networks cannot be distinguished by manipulating z; they are TS-equivalent with respect to this manipulation. Of course, TS-equivalence reduces to equivalence if no variables are manipulated.

DETERMINING A SEQUENCE OF EXPERIMENTS

Increasing the number of observations may contribute little to the full identification of the underlying causal network, due to network equivalence as described above. On the other hand, even a few additional experimental data can clarify underlying causal relationships. The following algorithm for learning a network by selecting specific manipulations starts with some observational data to derive a first set of highly scoring networks. It then repeatedly asks for manipulation of specific variables and for the data obtained from such experiments. It stops when the limit is reached for the number of manipulations the experimenter is willing to perform. In each round the algorithm selects manipulations that minimize a loss function on the information provided as described below. Our active learning algorithm is outlined in Figure 2.

The heuristic for variable selection is based on the following reasoning. Starting from a partition of graphs into equivalence classes based on observational data alone, each additional manipulation refines this partition by subpartitioning it into further TS-equivalence classes (this follows from the definition of TS-equivalence). Figure 3 shows an example of a single equivalence class that is partitioned into TS-equivalence subclasses. All networks in Figure 3b are indistinguishable based

input: observational data D with N_o samples,
limit on number of experiments,
further experimental data as requested
output: sequence of variables to manipulate

while limit not reached and
variables not yet manipulated exist **do**

learn TS-equivalence classes from D
keep K classes with highest probability
foreach variable a not yet manipulated **do**
$L_a \leftarrow$ expected loss $L(a, D)$ of
manipulation a given
current data D
select a with L_a minimum
output a
$D_a \leftarrow N_e$ new samples after manipulating a
$D \leftarrow D \cup D_a$ // update data

Fig. 2. Generation of a sequence of manipulations for active learning. The expected loss $L(a, D)$ is described in the text.

on observational data alone. Manipulating node C (Fig. 3c) results in more and smaller TS-equivalence subclasses than manipulating node A (Fig. 3d).

Consequently, the strategy is to pick a variable which splits the TS-equivalence class containing the true network into many subclasses of roughly equal size. Since the true network and the equivalence class containing it are unknown the choice must be based on a decision theoretic calculation. If a class is always split into a constant number of subclasses then we need a number of steps proportional to the logarithm of the class size for full identification of edge orientations. We want to minimize the number of steps and therefore adopt the logarithm of the class size as a loss function below. A decision which node to manipulate is then based on minimizing this loss function. Of course, this is only a heuristic, since the number of subclasses is not necessarily the same for each class.

At any stage, the K most likely TS-equivalence classes $E^i, i = 1, \ldots, K$, are considered. Owing to equivalence, all graphs in a class E^i have the same probability p_i. Manipulation of a particular gene a that has not yet been manipulated results in a subpartitioning of each E^i into TS-equivalence subclasses $E^i_j, j = 1, \ldots, n_i$. Let the index of the TS-equivalence class containing a particular graph g be $i(g)$ and that of the subclass within $E^{i(g)}$ containing g be $j(g)$. Note that the probability $p(g \mid D)$ of g is then $p_{i(g)}$. If g is the true network, the loss associated with manipulation of a is $\log |E^{i(g)}_{j(g)}|$, as motivated above. Since the true network is unknown the best we can do is to calculate the expected loss—based on posterior graph probabilities p_i obtained from

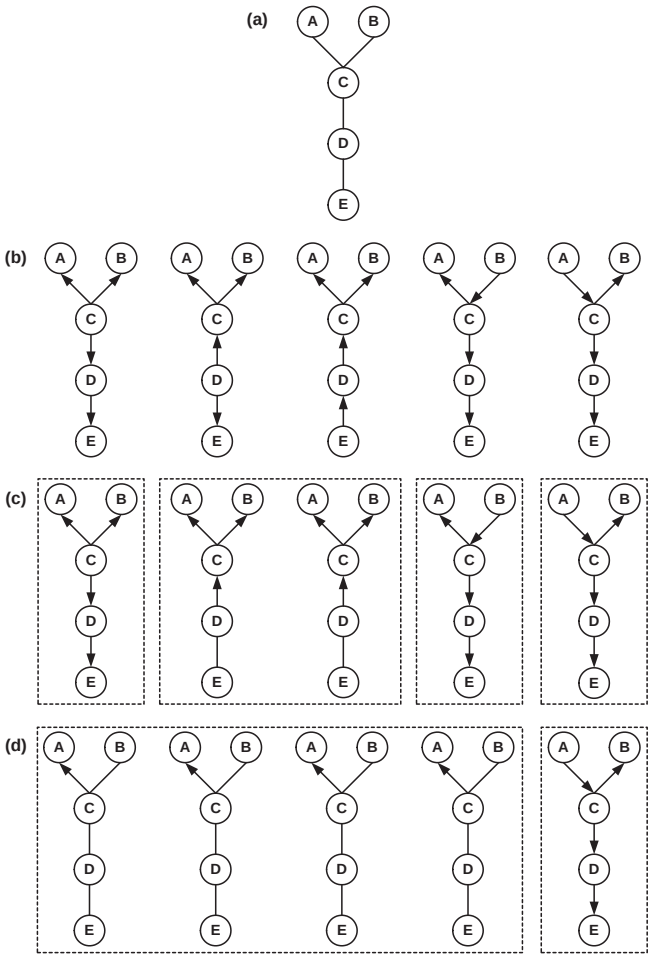

Fig. 3. A schematic representation of TS-equivalence. (**a**) A PDAG. (**b**) Equivalent networks that correspond to the PDAG in (a). (**c**) TS-equivalence classes after C is manipulated in the corresponding graphs in (a). The effect of manipulation is that all the incoming and outgoing edges in C are compelled; compelled edges are then propagated. (**d**) TS-equivalence classes after A is manipulated in the corresponding graphs in (a).

current experimental and observational data D:

$$L(a, D) = E_g\left(\log\left|E^{i(g)}_{j(g)}\right|\right) = \sum_g p_{i(g)} \log\left|E^{i(g)}_{j(g)}\right|$$

$$= \sum_{i=1}^{K}\sum_{j=1}^{n_i} p_i \left|E^i_j\right| \log\left|E^i_j\right| \qquad (1)$$

The algorithm records the manipulation a that minimizes the expected loss and enhances the current data D by adding data D_a obtained after performing a. Experiments in which two or more variables are manipulated simultaneously can be evaluated in a similar fashion. A sequential importance sampling algorithm described in the methods section is used to estimate the size of large TS-equivalence classes.

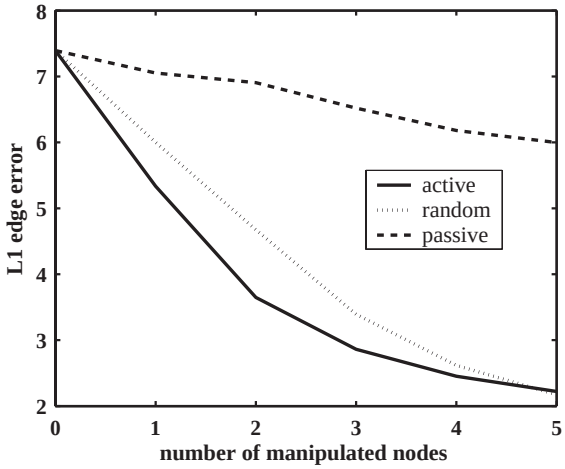

Fig. 4. Edge error E(P) of learning strategies for cancer network: active learning (solid line), learning by random manipulations (dotted line), and passive learning with observations only (dashed line). Lines are averages over 100 runs with randomly generated conditional probability tables.

RESULTS

We compared results obtained by our algorithm with results of active learning algorithms applied to the cancer network discussed in Tong and Koller (2001) and Murphy (2001). We also compared its performance on large sets of randomly generated networks with that of randomly selected manipulations. Finally, we examined the accuracy of the sequential importance sampling algorithm for the estimation of the size of equivalence classes.

Cancer network

For direct comparison with the algorithms of Tong and Koller (2001) and Murphy (2001), we ran our active learning algorithm on the cancer network which consists of five nodes with discrete domains. We started with $N_o = 20$ observational data and for each manipulated node we sampled $N_e = 10$ experimental data, to make our results comparable to those in Tong and Koller (2001). The performance of the active learning algorithm is evaluated based on the L_1 edge error as described in Tong and Koller (2001) and Murphy (2001):

$$E(P) = \sum_i \sum_j I_{G^*}(X_i \to X_j)[1 - P(X_i \to X_j)]$$
$$+ I_{G^*}(X_i \leftarrow X_j)[1 - P(X_i \leftarrow X_j)]$$
$$+ I_{G^*}(X_i \perp X_j)[1 - P(X_i \perp X_j)] \qquad (2)$$

where $I_{G^*}(\cdot)$ is 1, if $(\cdot)$ holds in the target network, G^* and 0 otherwise. Probabilities for features such as edge orientations are obtained by summing the probabilities of networks containing the feature.

Figure 4 shows that our active learning algorithm outperforms the random learning algorithm ($P < 0.0004$, Hotelling

T^2 test). Visual inspection of the corresponding Figure 1a in Tong and Koller (2001) suggests that our approach performs better than the algorithm presented there [note that in Tong and Koller (2001) the algorithm is aided by a selection of potential parents that includes the true parents]. Similarly, an implementation of the algorithm in Murphy (2001) gives results which seem less well separated from a random choice than the results of our algorithm (data not shown).

Performance on random networks

A total of 100 target networks with a fixed number of random edges were constructed. Each node was connected to at least one other randomly selected node to prevent isolated nodes. Further, edges were added or removed randomly until an acyclic graph was obtained with the number of edges in a prescribed range. For discrete domains conditional probability tables were generated randomly for its binary variables. For continuous domains normal variables with zero mean, random error variances between 0.1 and 0.5 and regression coefficients of 1 for parent nodes were generated. $N_o = 100$ observational cases were sampled from each network. $N_e = 10$ cases were sampled from each network after each manipulation. Variables were manipulated according to the sequence prescribed by the active learning algorithm of Figure 2. Manipulation of a discrete variable consisted in setting it to zero in one-half of the cases and to one in the other. Manipulation of a continuous variable consisted in setting all regression coefficients for parent nodes to 0. Calculations of manipulations for one network of 60 nodes took 2.2 h on a Pentium 4, 2.5 GHz. Most of the time was spent on learning networks, only 1 min was spent on calculating the sequence of manipulations itself.

A single combined network was obtained from the set of the K highest-scoring networks for comparison with the target network in the following way. Probabilities for the three possible causal relationships were calculated for each pair of variables by summing the probabilities of networks containing the relationship. Edges and edge directions with the highest probabilities were taken into the combined network. The combined DAG was converted into a PDAG in which edges attached to manipulated nodes were set as compelled. This PDAG was compared to the PDAG of the target network. A relation between two nodes was counted as correct if the edge was either missing in both PDAGs, or if present was either undirected or oriented in the same direction.

We compared the *active learning* algorithm of Figure 2 with a *random learning* algorithm which is similar except that variables for manipulation are selected randomly. To estimate the effect of sampling errors we applied an *informed learning* algorithm in which variables for manipulation are calculated from the PDAG of the target network directly.

For discrete domains, the algorithm was evaluated on networks with 10 nodes and 20 to 22 edges. Bayesian learning for discrete cases with a large number of nodes is computationally expensive. For continuous domains, the algorithm was evaluated on networks with 60 nodes and 120–125 edges. We used a penalty for the number of edges as suggested by Heckerman *et al*. (1995) to prevent overprediction in the Bayesian learning procedure. The size of equivalence classes was estimated by the importance sampling algorithm described in the methods section (sample size 1000).

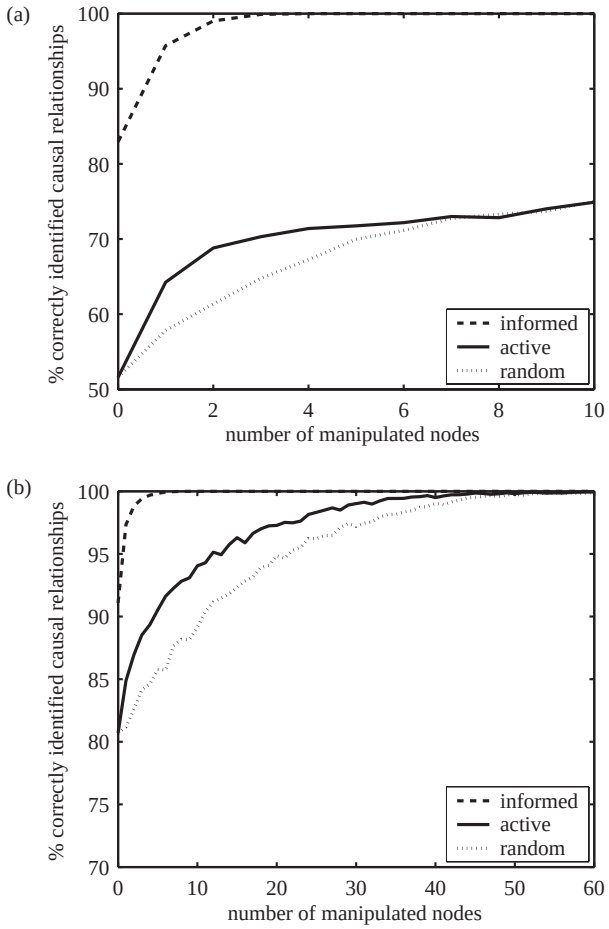

Fig. 5. The percentage of correctly identified causal relationships for (**a**) discrete and (**b**) continuous domains. Informed, active, and random learning are indicated by dashed, solid, and dotted lines respectively.

Figure 5 shows that the active learning algorithm outperforms random learning. As the figure shows, the difference between the active and random learning is most pronounced for small numbers of manipulations. At the point where almost all nodes are manipulated their order becomes less important. The difference between informed and active learning is due to sampling error. If the learned network is very different from the target network, the node suggested for manipulation by the active learning algorithm is no longer informative. Comparing Figures 5a and b it is seen that the performance is always better for continuous variables than for discrete variables on the same number of samples.

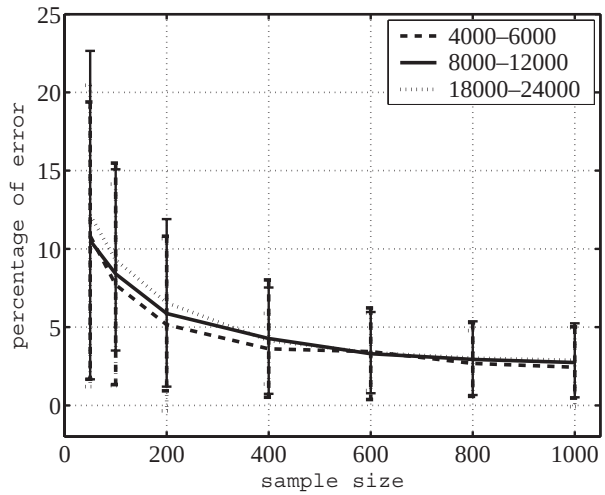

Fig. 6. The average percentage of error between estimated and real size of equivalence classes for 2000 networks with 30 nodes. The results have been split into three groups (4000–6000, 8000–12 000, 18 000–24 000) according to the real size of the equivalence class.

Estimation of the size of equivalence classes

We evaluated the sequential importance sampling algorithm for estimating the size of equivalence classes described in the methods section below. Figure 6 shows the percentage error between the estimated and the real size of an equivalence class in a simulation. Shown are the averages and error bars for one standard deviation over 2000 random networks each comprising 30 nodes. In addition, the results are grouped according to the real size of the equivalence class. Surprisingly, the error in estimation is almost identical for all three groups, indicating that the relative approximation error is independent of the size of the class.

DETAILS OF METHODS

Enumeration of TS-equivalent networks

In a naive approach to enumerating the equivalence class of a PDAG all possible directions of reversible edges would be considered and networks with a new v-structure or a cycle discarded. More efficient algorithms propagate the orientation of edges taking advantage of absent v-structures as in the algorithm of Figure 7. For an illustration of the difference between the two approaches see Figure 8.

It is easy to see that the algorithm in Figure 7 is correct. The step that needs some explanation is why only triangles are checked for cycles. If a cycle that is not a triangle is created at some point, a cyclic triangle will be detected later and the algorithm will backtrack. The reason for this is that if there is an edge that shortcuts the cycle, a chord, no matter how this chord is directed a smaller cycle results and we can use induction. If the cycle has no chord and is not a triangle then any DAG from the equivalence class must contain converging

input: DAG G
output: TS-equivalence class of G

convert DAG into PDAG
assume ordering of reversible edges
call directEdge(e) for first reversible edge e

function directEdge $(x - y)$
  if try$(x \to y)$ returns **success then**
    if no reversible edge **then**
      output network and **return**
    else
      call directEdge(e) for next
      reversible edge e
      reset all edges directed in try$(x \to y)$
  repeat **if** statement with try$(x \leftarrow y)$

function try$(u \to v)$
  if $u \to v$ creates 3-cycle, **return fail**
  direct $u - v$ as $u \to v$
  foreach w with $v - w$ reversible
      and u, w not connected **do**
    if try$(v \to w)$ returns, **then return fail**
  return success

Fig. 7. Enumeration algorithm for the equivalence class of a DAG

arcs which form a v-structure. The v-structure would be part of the PDAG, and the problem of a cycle would not arise in the first place. The latter observation also provides a reason why there is no new v-structure created: this could only happen during a recursive call in the *try* subroutine and would require that there is a chordless cycle of unoriented edges.

Although this still leaves the possibility that the algorithm detects a cycle at a very late stage and has to backtrack, this is rarely observed in practice if reversible edges are ordered according to a depth first enumeration of triangles. Another scheme for enumerating all graphs of an equivalence class is based on simplicial nodes without outgoing edges (see Dor and Tarsi, 1992).

The above algorithm for enumerating networks from an equivalence class is easily converted into one enumerating networks from a TS-equivalence class defined by a DAG and a set of manipulated variables. TS-equivalent networks have the same set of parents (and consequently children) for all manipulated variables. To enumerate TS-equivalent networks we add a dummy parent to each manipulated variable. This converts the manipulated variable into a v-structure with respect to each parent and the dummy node. Hence, when constructing the PDAG all these orientations around a manipulated variable are preserved. For nodes with no parents we need two dummy nodes to enforce the directions from the node to its children.

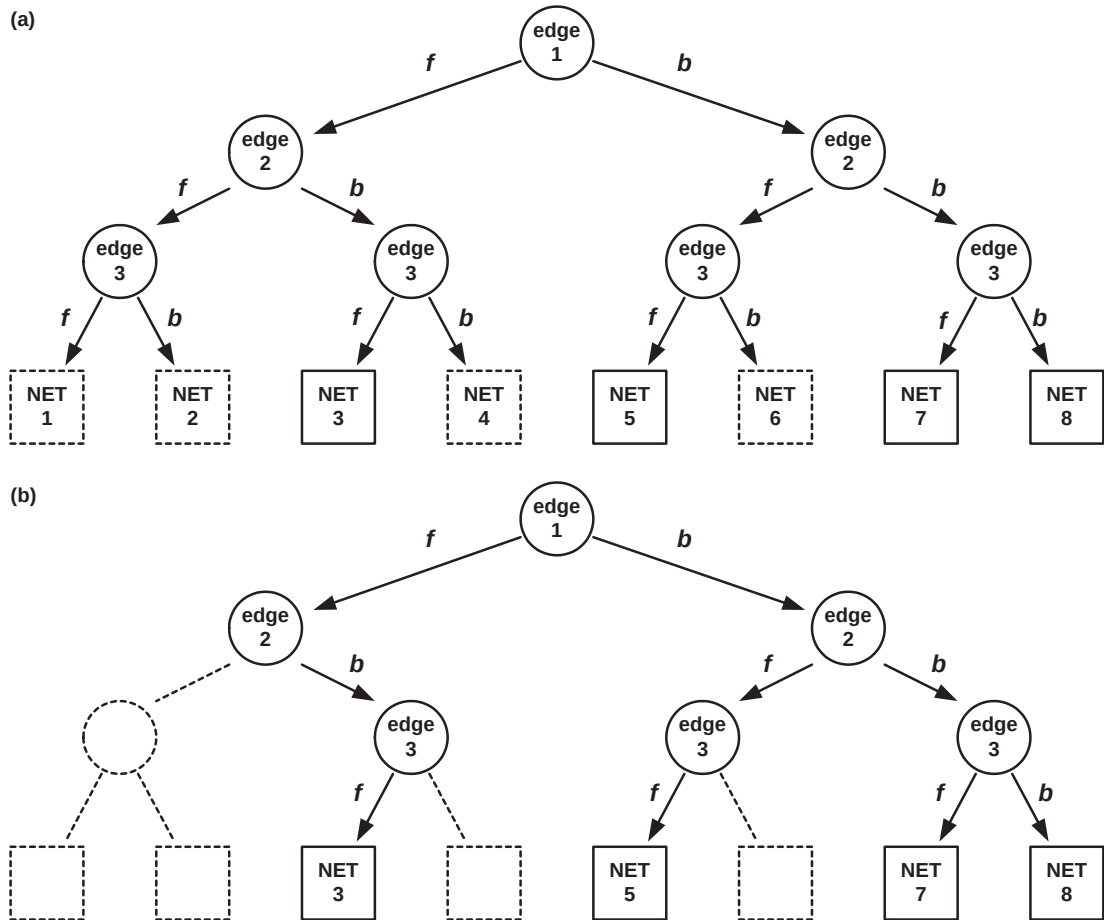

Fig. 8. A tree representation of the naive algorithm (**a**) and the algorithm of Figure 7 (**b**) for enumerating all equivalent networks, applied to the network in Figure 1. Internal nodes (circles) represent edges, leaves (squares) full orientations. Each edge can have one of two directions: f (nodes in alphabetical order) and b (reverse order). Valid orienting moves are indicated by full arrows, invalid moves by dashed lines.

Estimating the number of TS-equivalent networks

To estimate the size of a large equivalence class we use the following sequential importance sampling scheme. Suppose we define a probability distribution for a random variable G and a function w so that the expected value of $w(G)$ is the size $|E|$ of an equivalence class E. We can then estimate the size of the class by sampling G and averaging $w(G)$. For this purpose we convert the algorithm of Figure 7 into a random algorithm that samples partially directed graphs g with weights $w(g)$.

We convert subroutine directEdge into a random subroutine directEdgeRandom. Subroutine *try* is called for each direction of edge $x - y$. If both calls return success one direction is chosen randomly with probability 1/2 and directEdgeRandom called on it. If only one call returns success directEdgeRandom is called on it. If both calls return fails the partially oriented graph generated so far is output with weight 0. On the other hand, should the algorithm succeed in producing a DAG g—which must be from equivalence class E—its weight is set to $w(g) = 2^d$, where d is the number of decisions made along the path to g. Note that the probability of generating g is $q(g) = 1/2^d$, so $w(g) = 1/q(g)$. For an example, see the tree in Figure 8b where d is the number of nodes with two possible children along the path to g. Hence, the weights of networks 3, 5, 7 and 8 are 2, 4, 8 and 8, respectively.

Calculating the expected weight we obtain

$$E(w) = \sum_g q(g)w(g) = \sum_g I(g \in E) = |E| \qquad (3)$$

where the summation is over all possible outcomes g of the above algorithm. I is an indicator function which is 1 for true statements and 0 for false ones. Consequently, the size of E can be estimated by sampling with the above random algorithm and by averaging over the resulting weights. In the above example, the average weight is 4, the size of the equivalence class.

Similarly, one can estimate the contribution of a large TS-equivalence class E^i to $L(a, D)$ in Equation (1). We sample graphs from E^i and for each sampled graph g we estimate the size of its TS-equivalence subclass $E^i_{j(g)}$ by the above sampling scheme. The expectation of $w(g)p_i \log|E^i_{j(g)}|$ is

$$\sum_g q(g)w(g)p_i \log|E^i_{j(g)}| = \sum_j p_i|E^i_j| \log|E^i_j|$$

the contribution of E^i to $L(a, D)$. The first sum is over all possible outcomes g of the sampling algorithm, and the second sum is over all equivalence subclasses E^i_j of E^i. Hence, to estimate the contribution of E^i all we need to do is to average $w(g)p_i \log|E^i_{j(g)}|$ for graphs g sampled from E^i.

DISCUSSION

For the time being, simulation results are almost the only way to evaluate approaches to active learning. We are currently applying the algorithm to microarray data, but have no objective measure to evaluate its performance in a realistic experimental setting. For simulated data we can show that our method performs significantly better than a random selection of manipulation experiments. Not surprisingly, data from continuous domains are much more informative than data from discrete domains. A disadvantage of the type Bayesian networks used here is that they only allow us to model linear dependencies. Preliminary results (not shown here) suggest that this might not be too severe a restriction when applied to the logarithm of gene expression levels. Linear relations between logarithmic values essentially capture multiplicative dependencies between co-regulating genes.

Our active learning approach based on expected loss—where loss is defined in terms of the size of TS-equivalence classes—has comparable or even better performance than algorithms based on the expected value of information. This suggests that the main contribution to active learning of Bayesian networks stems from the structure of TS-equivalence classes.

A further finding is that, for the networks investigated here, the gain in additional information by interventions is highest for the first few interventions. This is encouraging for it suggests that ambiguities in causal relationships arising from observational data can often be resolved with just a few well chosen experiments.

ACKNOWLEDGEMENTS

We would like to thank Kevin Murphy for providing us with the code for his active learning algorithm and the referees for helpful comments. I.P. is supported by a studentship from the MRC and a grant from the BBSRC. L.W. acknowledges an equipment grant from the Royal Society and a grant from the Wellcome Trust.

REFERENCES

Chickering,D. (1995) A Transformational characterization of equivalent Bayesian network structures. *Proceedings of Eleventh Conference on Uncertainty in Artificial Intelligence*. Montreal 8798. Morgan Kaufman, San Francisco, CA. 87–98.

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

Cooper,G. and Yoo,C. (1999) Causal discovery from a mixture of experimental and observational data. *Proceedings of Fifteenth Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, CA. 116–125.

Dor,D. and Tarsi,M. (1992) A simple algorithm to construct a consistent extension of a partially oriented graph. *Technical Report R-185*, Cognitive Systems Laboratory, Computer Science Department, UCLA.

Friedman,N., Goldszmidt,M. and Wyner,A. (1999) Data analysis with Bayesian networks: a bootstrap approach. *Proceedings of Fifteenth Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, CA. 196–205.

Friedman,N., Linial,M., Nachman,I. and Pe'er,D. (2000) Using Bayesian networks to analyze expression data. *J. Comput. Biol.*, **7**, 601–620.

Friedman,N. and Koller,D. (2000) Being Bayesian about network structure. *Proceedings of Sixteenth Conference on Uncertainty in Artificial Intelligence*. Stanford, CA. 201–210.

Geiger,D. and Heckerman,D. (1994) Learning Gausian networks. *Proceedings of Tenth Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann San Francisco, CA. 235–243.

Hartemink,A., Gifford,S., Jaakkola,J. and Young,R. (2001) Using graphical models and genomic expression data to statistically validate models of genetic regulatory networks. *Pac. Symp. Biocomp.*, **6**, 422–433.

Heckerman,D. (1995) A Tutorial on learning Bayesian networks. *Technical Report MSR-TR-95-06*, Microsoft Research, Redmond, WA.

Heckerman,D. and Geiger,D. (1995) Likelihoods and parameter priors for Bayesian. *Technical Report MSR-TR-95-54*, Microsoft Research, Redmond, WA.

Heckerman,D., Geiger,D. and Chickering,D. (1995) Learning Bayesian networks: the combination of knowledge and statistical data. *Mach. Learning*, **20**, 197–243.

Ideker,T., Thopsson,V. and Karp,R. (2000) Discovery of regulatory interactions through perturbation: inference and experimental design. *Pac. Symp. Biocomp.*, **5**, 302–313.

Liu,J., Chen,R. and Logvinenko,T. (2001) A theoretical framework for sequential importance sampling and resampling. In Doueet,A., de Freitas,J.F.G. and Gordon, N. (eds), Cambridge University Press, *Sequential Monte Carlo Methods in Practice*, New York, pp. 863–869.

Meek,C. (1995) Causal inference and causal explanation with background knowledge. *Proceedings of Eleventh Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, San Mateo, CA. 403–410.

Melancon,G., Dutour,I. and Bousquet-Melou,M. (2000) Random generation of dags for graph drawing. INSROOS Centre for Mathematics and Computer Sciences, Amsterdam.

Murphy,K. (2001) Active learning of causal Bayes net structure. *Technical Report*. University of California, Berkeley, CA.

Pearl,J. (2000) *Causality: Models, Reasoning and Inference*. Cambridge University Press, NY.

Segal,E., Taskar,B., Gasch,A., Friedman,N. and Koller,D. (2001) Rich probabilistic models for gene expression. *Bioinformatics*, **17**, S243–S252.

Spirtes,P., Glymour,C. and Scheines,R. (2001) *Causation, Prediction, and Search*. Springer-Verlag, NY.

Tian,J. and Pearl,J. (2001) Causal discovery from changes: a Bayesian approach. *Technical Report (R-285), UCLA Cognitive Systems Laboratory*.

Tong,S. and Koller,D. (2001) Active learning for structure in Bayesian networks. *Seventeenth International Joint Conference on Artificial Intelligence*. Seattle, Washington, 863–869.

Verma,T. and Pearl,J. (1990) Equivalence and synthesis of causal models. *Proceedings of Sixth Conference on Uncertainty in Artificial Intelligence*.

APPENDIX: TS-EQUIVALENT SCORES FOR CONTINUOUS DOMAIN

We extend the score for Bayesian networks on discrete domains with manipulations as described in Cooper and Yoo (1999) to continuous domains. Geiger and Heckerman (1994) derived the Bayesian Gaussian likelihood equivalent (BGe) metric:

$$p(D, G \mid \xi) = p(G \mid \xi) \prod_{i=1}^{n} \frac{p(D^{\pi_{x_i}, x_i} \mid G_c, \xi)}{p(D^{\pi_{x_i}} \mid G_c, \xi)} \qquad (4)$$

$D^{\pi_{x_i}, x_i}$ is the database D restricted to the variables π_{x_i} and x_i and G_c is a complete Gaussian belief network.

If a variable is manipulated, then its value is independent of all other variables, and thus no incoming arcs are incident to that node. $U' \subset U$ is a set of such variables in cases $D' \subset D$. We assume that the unconditional mean and conditional variance for each node remains the same. Given this assumption, the parameters of each node that do not belong to U' can be estimated from the whole dataset D, while the parameters of each node belonging to U' must be estimated from two different subsets. The first corresponds to the experiments where the value of the manipulated node was influenced by its parents, and the second set corresponds to the dataset where the manipulated node did not have any parent. Thus, the BGe metric becomes:

$$\begin{aligned} p(D, G \mid \xi) \quad = \quad & p(G \mid \xi) \prod_{i \notin U'} \frac{p(D^{\pi_{x_i}, x_i} \mid G_c, \xi)}{p(D^{\pi_{x_i}} \mid G_c, \xi)} \\ & \times \prod_{i \in U'} \frac{p((D \setminus D_i)^{\pi_{x_i}, x_i} \mid G_c, \xi)}{p((D \setminus D_i)^{\pi_{x_i}} \mid G_c, \xi)} \\ & \times p(D_i^{x_i} \mid G_c, \xi) \end{aligned}$$

where $D_i \subset D'$ denotes the dataset in which node x_i was manipulated.