Modelling Gene Expression Data using Dynamic Bayesian Networks

Kevin Murphy and Saira Mian
Computer Science Division, University of California
Life Sciences Division, Lawrence Berkeley National Laboratory
Berkeley, CA 94720
Tel (510) 642 2128, Fax (510) 642 5775
murphyk@cs.berkeley.edu, smian@lbl.gov

Abstract

Recently, there has been much interest in reverse engineering genetic networks from time series data. In this paper, we show that most of the proposed discrete time models — including the boolean network model [Kau93, SS96], the linear model of D'haeseleer et al. [DWFS99], and the nonlinear model of Weaver et al. [WWS99] — are all special cases of a general class of models called Dynamic Bayesian Networks (DBNs). The advantages of DBNs include the ability to model stochasticity, to incorporate prior knowledge, and to handle hidden variables and missing data in a principled way. This paper provides a review of techniques for learning DBNs. **Keywords:** Genetic networks, boolean networks, Bayesian networks, neural networks, reverse engineering, machine learning.

1 Introduction

Recently, it has become possible to experimentally measure the expression levels of many genes simultaneously, as they change over time and react to external stimuli (see e.g., [WFM+98, DLB97]). In the future, the amount of such experimental data is expected to increase dramatically. This increases the need for automated ways of discovering patterns in such data. Ultimately, we would like to automatically discover the structure of the underlying causal network that is assumed to generate the observed data.

In this paper, we consider learning stochastic, discrete time models with discrete or continuous state, and hidden variables. This generalizes the linear model of D'haeseleer et al. [DWFS99], the nonlinear model of Weaver et al. [WWS99], and the popular boolean network model [Kau93, SS96], all of which are deterministic and fully observable.

The fact that our models are stochastic is very important, since it is well known that gene expression is an inherently stochastic phenomenon [MA97]. In addition, even if the underlying system were deterministic, it might *appear* stochastic due to our inability to perfectly measure all the variables. Hence it is crucial that our learning algorithms be capable of handling noisy data. For example, suppose the underlying system really is a boolean network, but that we have noisy observations of some of the variables. Then the data set might contain inconsistencies, i.e., there might not be any boolean network which can model it. Rather than giving up, we should look for the most *probable* model given the data; this of course requires that our model have a well-defined probabilistic semantics.

The ability of our models to handle hidden variables is also important. Typically, what is measured (usually mRNA levels) is only one of the factors that we care about; other ones include cDNA levels, protein levels, etc. Often we can model the relationship between these factors, even if we cannot measure their values. This prior knowledge can be used to constrain the set of possible models we learn.

The models we use are called Bayesian (belief) Networks (BNs) [Pea88], which have become the method of choice for representing stochastic models in the UAI (Uncertainty in Artificial Intelligence) community. In Section 2, we explain what BNs are, and show how they generalize the boolean network model [Kau93, SS96], Hidden Markov Models [DEKM98], and other models widely used in the computational biology community. In Sections 3 to 7, we review various techniques for learning BNs from data, and show how REVEAL [LFS98] is a special case of such an algorithm. In Section 8, we consider BNs with continuous (as opposed to discrete) state, and discuss their relationship to the the linear model of D'haeseleer et al. [DWFS99], the nonlinear model of Weaver et al. [WWS99], and techniques from the neural network literature [Bis95].

2 Bayesian Networks

BNs are a special case of a more general class called graphical models in which nodes represent random variables, and the lack of arcs represent conditional independence assumptions. Undirected graphical models, also called Markov Random Fields (MRFs; see e.g., [WMS94] for an application in biology), have a simple definition of independence: two (sets of) nodes A and B are conditionally independent given all the other nodes if they are separated in the graph. By contrast, directed graphical models (i.e., BNs) have a more complicated notion of independence, which

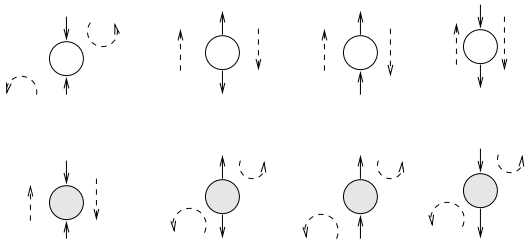

Figure 1: The Bayes-Ball algorithm. Two (sets of) nodes A and B are conditionally independent (d-separated [Pea88]) given all the others if and only if there is no way for a ball to get from A to B in the graph. Hidden nodes are nodes whose values are not known, and are depicted as unshaded; observed nodes are shaded. The dotted arcs indicate direction of flow of the ball. The ball cannot pass through hidden nodes with convergent arrows (top left), nor through observed nodes with any outgoing arrows. See [Sha98] for details.

takes into account the directionality of the arcs (see Figure 1). Graphical models with both directed and undirected arcs are called chain graphs.

In a BN, one can intuitively regard an arc from A to B as indicating the fact that A "causes" B. (For a more formal treatment of causality in the context of BNs, see [HS95].) Since evidence can be assigned to any subset of the nodes (i.e., any subset of nodes can be observed), BNs can be used for both causal reasoning (from known causes to unknown effects) an diagnostic reasoning (from known effects to unknown causes), or any combination of the two. The inference algorithms which are needed to do this are briefly discussed in Section 5.1. Note that, if all the nodes are observed, there is no need to do inference, although we might still want to do learning.

In addition to causal and diagnostic reasoning, BNs support the powerful notion of "explaining away": if a node is observed, then its parents become dependent, since they are rival causes for explaining the child's value (see the bottom left case in Figure 1.) In contrast, in an undirected graphical model, the parents would be independent, since the child separates (but does not d-separate) them.

Some other important advantages of directed graphical models over undirected ones include the fact that BNs can encode deterministic relationships, and that it is easier to learn BNs (see Section 3) since they are separable models (in the sense of [Fri98]). Hence we shall focus exclusively on BNs in this paper. For a careful study of the relationship between directed and undirected graphical models, see [Pea88, Whi90, Lau96].[1]

[1]It is interesting to note that much of the theory underlying graphical models involves concepts such as chordal (triangulated) graphs [Gol80], which also arise in other areas of computational biology, such as evolutionary tree construction (perfect phylogenies) and physical mapping (interval graphs).

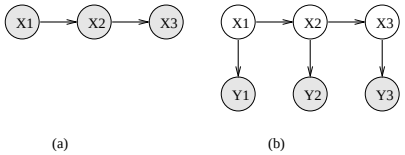

Figure 2: (a) A Markov Chain represented as a Dynamic Bayesian Net (DBN). (b) A Hidden Markov Model (HMM) represented as a DBN. Shaded nodes are observed, non-shaded nodes are hidden.

2.1 Relationship to HMMs

For our first example of a BN, consider Figure 2(a). We call this a *Dynamic* Bayesian Net (DBN) because it represents how the random variable X evolves over time (three time slices are shown). From the graph, we see that X_{t+1} is independent of X_{t-1} given X_t (since X_t blocks the only path for the Bayes ball between X_{t-1} and X_{t+1}). This, of course, is the (first-order) Markov property, which states that the future is independent of the past given the present.

Now consider Figure 2(b). X_t is as before, but is now *hidden*. What we observe at each time step is Y_t, which is another random variable whose distribution depends on (and only on) X_t. Hence this graph captures all and only the conditional independence assumptions that are made in a Hidden Markov Model (HMM) [Rab89].

In addition to the graph structure, a BN requires that we specify the Conditional Probability Distribution (CPD) of each node given its parents. In an HMM, we assume that the hidden state variables X_t are discrete, and have a distribution given by $\Pr(X_t = j|X_{t-1} = i) = T_t(i,j)$. ($T_t$ is the transition matrix for time slice t.) If the observed variables Y_t are discrete, we can specify their distribution by $\Pr(Y_t = j|X_t = i) = O_t(i,j)$. ($O_t$ is the observation matrix for time slice t.) However, in an HMM, it is also possible for the observed variables to be Gaussian, in which case we must specify the mean and covariance for each value of the hidden state variable, and each value of t: see Section 8.

As a (hopefully!) familiar example of HMMs, let us consider the way that they are used for aligning protein sequences [DEKM98]. In this case, the hidden state variable can take on three possible values, $X_t \in \{D, I, M\}$, which represent delete, insert and match respectively. In protein alignment, the t subscript does not refer to time, but rather to position along a static sequence. This is an important difference from gene expression, where t really does represent time (see Section 3).

The observable variable can take on 21 possible values, which represent the 20 possible amino acids, plus the gap alignment character "-". The probability distribution over these 21 values depends on the current position t and of the current state of the system, X_t. Thus the distribution $\Pr(Y_t|X_t = M)$ is the profile for position t, $\Pr(Y_t|X_t = I)$ is the ("time"-invariant) "background" distribution, and $\Pr(Y_t = '' - ''|X_t) = 1.0$ if $X_t = D$ and is 0.0 if $X_t \neq D$, for all t.

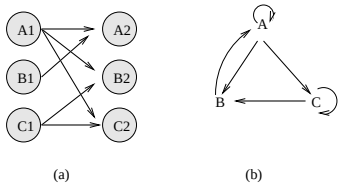

Figure 3: (a) A DBN, (b) Represented as a boolean network.

In an HMM, "learning" means finding estimates of the parameters T_t and O_t (see Section 4.1); the *structure* of the model is fixed (namely, it is Figure 2(b)).

2.2 Relationship to boolean networks

Now consider the DBN in Figure 3(a). (We only show two time slices of the DBN, since the structure repeats.) Just as an HMM can be thought of as a single stochastic automaton, so this can be thought of as a network of four interacting stochastic automata: node A_t represents the state of the A automaton at time t, for example; its next state is determined by its previous state, and the previous state of B.

The interconnectivity between the automata can also be represented as in Figure 3(b). The presence of directed cycles means that this is not a BN; rather, an arc from node i to node j means $A(i,j) = 1$, where A is the inter-slice adjacency matrix.

In addition to connections *between* time slices, we can also allow connections *within* a time slice (intra-slice connections). This can be used to model co-occurrence effects, e.g., gene B_{t-1} causes A_t to turn on if and only if C_t is also turned on. Since this relationship between A_t and C_t is not causal, it is more natural to model this with an undirected arc; the resulting model would therefore be a chain graph, which we don't consider in this paper.

The transition matrix of each automaton (e.g., $\Pr(A_t|A_{t-1}, B_{t-1})$) is often called a CPT (Conditional Probability *Table*), since it represents the CPD in tabular form. If each row of the CPT only contains a single non-zero value (which must therefore be 1.0), then the automaton is in fact a deterministic automaton. If all the nodes (automata) are deterministic and have two states, the system is equivalent to a boolean network [Kau93, SS96].

In a boolean network, "learning" means finding the best structure, i.e., inter-slice connectivity matrix (see Section 6). Once we know the correct structure, it is easy to figure out what logical rule each node is using, e.g., by exhaustively enumerating them all and finding the one that fits the data.

2.3 Stochastic boolean networks

We can define a *stochastic* boolean network model by modifying the CPDs. For example, a popular CPD in the UAI community is the noisy-OR model [Pea88]. This is just like an OR gate, except that there is a certain probability q_i that the i'th input will be flipped from 1 to 0. For example, if

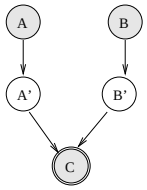

Figure 4: The noisy-OR gate modelled as a BN. A' is a noisy version of A, and B' is a noisy version of B. C is a deterministic OR of A' and B' (indicated by the double ring). Shaded nodes are observed, non-shaded nodes are hidden.

$C = \text{noisyor}(A, B)$, we can represent its CPD in tabular form as follows:

A	B	$\Pr(C=0)$	$\Pr(C=1)$
0	0	1.0	0.0
1	0	q_1	$1-q_1$
0	1	q_2	$1-q_2$
1	1	$q_1 q_2$	$1-q_1 q_2$

Note that this CPT has 2^{K+1} entries, where K is the number of parents (here, $K = 2$), but that there are constraints on these entries. For a start, the requirement that $\Pr(C = 0|A, B) + \Pr(C = 1|A, B) = 1.0$ for all values of A and B eliminates one of the columns, resulting in 2^K entries. However, the entries in the remaining columns are themselves functions of just K free parameters, $q_1, \ldots, q_K$. Hence this is called an implicit or parametric distribution, and has the advantage that less data is needed to learn the parameter values (see Section 3). By contrast, a full multinomial distribution (i.e., an unconstrained CPT) would have 2^K parameters, but could model arbitrary interactions between the parents.

The intepretation of the noisy-OR gate is that each input is an independent cause, which may be inhibited. This can be modelled by the BN shown in Figure 4. It is common to imagine that one of the parents is permanently on, so that the child can turn on "spontaneously". This "leak" node can be used as a catch-all for all other, unmodelled causes.

It is straightforward to generalize the noisy-OR gate to non-binary variables and other functions such as AND [Hen89, Sri93, HB94]. It is also possible to loosen the assumption that all the causes are independent [MH97].

Another popular compact representation for CPDs in the UAI community is a decision tree [BFGK96]. This is a stochastic generalization of the concept of canalyzing function [Kau93], popular in the boolean networks field. (A function is canalyzing if at least one of its inputs has the property that, when it takes a specific value, the output of the function is independent of all the remaining inputs.)

2.4 Comparison of DBNs and HMMs

Note that we can convert the model in Figure 3(a) into the one in Figure 2(a), by defining a new variable whose state space is the cross product of the original variables, $X_t =$

(A_t, B_t, C_t), i.e., by "collapsing" the original model into a chain. But now the conditional independence relationships are "buried" inside the transition matrix. In particular, the entries in the transition matrix are products of a smaller number of parameters, i.e.,

$$\Pr(X_t|X_{t-1}) = \Pr(A_t|A_{t-1}, B_{t-1}) \times \Pr(B_t|A_{t-1}, C_{t-1}) \times \Pr(C_t|A_{t-1}, C_{t-1})$$

So 0s in the transition matrix do *not* correspond to absence of arcs in the model. Rather, if $T(i,j) = 0$, it just means the automaton cannot get from state i to state j — this says nothing about the connections between the underlying state *variables*. Thus Markov chains and HMMs are not a good representation for sparse, discrete models of the kind we are considering here.

2.5 Higher-order models

We have assumed that that our models have the first-order Markov property, i.e., that there are only connections between adjacent time slices. This assumption is without loss of generality, since it is easy to convert a higher-order Markov chain to a first-order Markov chain, simply by adding new variables (called lag variables) which contain the old state, and clamping the transition matrices of these new variables to the identity matrix. (Note that doing this for an HMM (as opposed to a DBN) would blow up the state space exponentially.)

2.6 Relationship to other probabilistic models

Although the emphasis in this section has been on models which are relevant to gene expression, we should remark that Bayesian Nets can also be used in evolutionary tree construction and genetic pedigree analysis. (In fact, the peeling (Elston-Stewart) algorithm was invented over a decade before being rediscovered in the UAI community.) Also, Bayesian networks can be augmented with utility and decision nodes, to create an influence diagram; this can be used to compute an optimal action or sequence of actions (optimal in the decision-theoretic sense of maximizing expected utility).This might be useful for designing gene-knockout experiments, although we don't pursue this issue here.

3 Learning Bayesian Networks

In this section, we discuss how to learn BNs from data. The problem can be summarized as follows.

Structure/ Observability	Method
Known, full	Sample statistics
Known, partial	EM or gradient ascent
Unknown, full	Search through model space
Unknown, partial	Structural EM

Full observability means that the values of all variables are known; partial observability means that we do not know the values of some of the variables. This might be because they cannot, in principle, be measured (in which case they are usually called hidden variables), or because they just happen to be unmeasured in the training data (in which case they are called missing variables). Note that a variable can be observed intermittently.

Unknown structure means we do not know the complete topology of the graph. Typically we will know some parts of it, or at least know some properties the graph is likely to have, e.g., it is common to assume a bound, K, on the maximum number of parents (fan-in) that a node can take on, and we may know that nodes of a certain "type" only connect to other nodes of the same type. Such prior knowledge can either be a "hard" constraint, or a "soft" prior.

To exploit such soft prior knowledge, we must use Bayesian methods, which have the additional advantage that they return a distribution over possible models instead of a single best model. Handling priors on model structures, however, is quite complicated, so we do not discuss this issue here (see [Hec96] for a review). Instead, we assume that the goal is to find a single model which maximizes some scoring function (discussed in Section 6.1). We will, however, consider priors on *parameters*, which are much simpler. In Section 8.1, when we consider DBNs in which all the variables are continuous, we will see that we can use numerical priors as a proxy for structural priors.

An interesting approximation to the full-blown Bayesian approach is to learn a mixture of models, each of which has different structure, depending on the value of a hidden, discrete, mixture node. This has been done for Gaussian networks [TMCH98] and discrete, undirected trees[2] [MJM98]; unfortunately, there are severe computational difficulties in generalizing this technique to arbitrary, discrete networks.

In this paper, we restrict our attention to learning the structure of the inter-slice connectivity matrix of the DBN. This has the advantage that we do not need to worry about introducing (directed) cycles, and also that we can use the "arrow of time" to disambiguate the direction of causal relations, i.e., we know an arc must go from X_{t-1} to X_t and not vice versa, because the network models the temporal evolution of a physical system. Of course, we still cannot completely overcome the problem that "correlation is not causation": it is possible that the correlation between X_{t-1} and X_t is due to one or more hidden common causes (or observed common effects, because of the explaining away phenomenon). Hence the models learned from data should be subject to experimental verification.[3]

[2] In one particularly relevant example, [MJM98] applies her algorithm to the problem of classifying DNA splice sites as intron-exon or exon-intron. When she examined each tree in the mixture to find the nodes which are most commonly connected to the class variable, she found that they were the nodes in the vicinity of the splice junction, and further, that their CPTs encoded the known AG/G pattern.

[3] [HMC97] discusses Bayesian techniques for learning causal (static) networks, and [SGS93] discusses constraint based techniques — but see [Fre97] for a critique of this approach. These techniques have mostly been used in the social sciences. One major advantage of molecular biology is that it is usually feasible

When the structure of the model is known, the learning task becomes one of parameter estimation. Although the focus of this paper is on learning structure, this calls parameter learning as a subroutine, so we start by discussing this case.

Finally, we mention that learning BNs is a huge subject, and in this paper, we only touch on the aspects that we consider most relevant to learning genetic networks from time series data. For further details, see the review papers [Bun94, Bun96, Hec96].

4 Known structure, full observability

We assume that the goal of learning in this case is to find the values of the parameters of each CPD which maximizes the likelihood of the training data, which contains S sequences (assumed to be independent), each of which has the observed values of all n nodes per slice for each of T slices. For notational simplicity, we assume each sequence is of the same length. Thus we can imagine "unrolling" a two-slice DBN to produce a (static) BN with T slices.

We assume the parameter values for all nodes are constant (tied) across time (in contrast to the assumption made in HMM protein alignment), so that for a time series of length T, we get one data point (sample) for each CPD in the initial slice, and $T-1$ data points for each of the other CPDs. If $S=1$, we cannot reliably estimate the parameters of the nodes in the first slice, so we usually assume these are fixed apriori. That leaves us with $N = S(T-1)$ samples for each of the remaining CPDs. In cases where N is small compared to the number of parameters that require fitting, we can use a numerical prior to regularize the problem (see Section 4.1). In this case, we call the estimates Maximum A Posterori (MAP) estimates, as opposed to Maximum Likelihood (ML) estimates.

Using the Bayes Ball algorithm (see Figure 1), it is easy to see that each node is conditionally independent of its non-descendants given its parents (indeed, this is often taken as the *definition* of a BN), and hence, by the chain rule of probability, we find that the joint probability of all the nodes in the graph is

$$P(X_1, \ldots, X_m) = \\ = \prod_i P(X_i | X_1, \ldots, X_{i-1}) = \prod_i P(X_i | \text{Pa}(X_i)),$$

where $m = n(T-1)$ is the number of nodes in the unrolled network (excluding the first slice[4]), $\text{Pa}(X_i)$ are the parents of node i, and nodes are numbered in topological order (i.e., parents before children). The normalized log-likelihood of the training set $L = \frac{1}{N} \log \Pr(D|G)$, where $D = \{D_1, \ldots, D_S\}$, is a sum of terms, one for each node:

$$L = \frac{1}{N} \sum_{i=1}^{m} \sum_{l=1}^{S} \log P(X_i | \text{Pa}(X_i), D_l). \quad (1)$$

We see that the log-likelihood scoring function *decomposes* according to the structure of the graph, and hence we can maximize the contribution to the log-likelihood of each node independently (assuming the parameters in each node are independent of the other nodes).

4.1 Parameter estimation

For discrete nodes with CPTs, we can compute the ML estimates by simple counting. As is well known from the HMM literature, however, ML estimates of CPTs are prone to sparse data problems, which can be solved by using (mixtures of) Dirichlet priors (pseudo counts).

Estimating the parameters of noisy-OR distributions (and its relatives) is harder, essentially because of the presence of hidden variables (see Figure 4), so we must use the techniques discussed in Section 5.

4.2 Choosing the form of the CPDs

In the above discussion, we assumed that we knew the form of the CPD for each node. If we are uncertain, one approach is to use a mixture distribution, although this introduces a hidden variable, which makes things more complicated (see Section 5). Alternatively, we can use a decision tree [BFGK96], or a table of parent values along with their associated non-zero probabilities [FG96], to represent the CPD. This can increase the number of free parameters gradually, from 1 to 2^k, where k is the number of parents.

5 Known structure, partial observability

When some of the variables are not observed, the likelihood surface becomes multimodal, and we must use iterative methods, such as EM [Lau95] or gradient ascent [BKRK97], to find a local maximum of the ML/MAP function. These algorithms need to use an inference algorithm to compute the expected sufficient statistics (or related quantity) for each node.

5.1 Inference in BNs

Inference in Bayesian networks is a huge subject which we will not go into in this paper. See [Jen96] for an introduction to one of the most commonly used algorithm (the junction tree algorithm). [HD94] gives a good cookbook introduction to the junction tree algorithm. [SHJ97] explains how the forwards-backwards algorithm is a special case of the junction tree algorithm, and might be a good place to start if you are familiar with HMMs. [Dec98] would be a good place to start if you are familiar with the peeling algorithm (although the junction tree approach is much more efficient for learning). [Mur99] discusses inference in BNs with discrete and continuous variables, and efficient techniques for handling networks with many observed nodes.

Exact inference in densely connected (discrete) BNs is computationally intractible, so we must use approximate methods. There are many approaches, including sampling meth-

to verify hypotheses experimentally.

[4] Since we are not trying to estimate the parameters of nodes in the initial slice, we have omitted their contribution to the overall probability, for simplicity.

ods such as MCMC [Mac98] and variational methods such as mean-field [JGJS98]. DBNs are even more computationally intractible in the sense that, even if the connections between two slices are sparse, correlations can arise over several time steps, thus rendering the unrolled network "effectively" dense. A simple approximate inference algorithm for the specific case of sparse DBNs is described in [BK98b, BK98a].

6 Unknown structure, full observability

We start by discussing the scoring function which we use to select models; we then discuss algorithms which attempt to optimize this function over the space of models, and finally examine their computational and sample complexity.

6.1 The objective function used for model selection

It is commonly assumed that the goal is to find the model with maximum likelihood. Often (e.g., as in the REVEAL algorithm [LFS98], and as in the Reconstructability Analysis (RA) or General Systems Theory community [Kli85, Kri86]) this is stated as finding the model in which the sum of the mutual information (MI) [CT91] between each node and its parents is maximal; in Appendix A, we prove that these objective functions are equivalent, in the sense that they rank models in the same order.

The trouble is, the ML model will be a complete graph, since this has the largest number of parameters, and hence can fit the data the best. A well-principled way to avoid this kind of over-fitting is to put a prior on models, specifying that we prefer sparse models. Then, by Bayes' rule, the MAP model is the one that maximizes

$$\Pr(G|D) = \frac{\Pr(D|G)\Pr(G)}{\Pr(D)} \tag{2}$$

Taking logs, we find

$$\log \Pr(G|D) = \log \Pr(D|G) + \log \Pr(G) + c$$

where $c = \log \Pr(D)$ is a constant independent of G. Thus the effect of the prior is equivalent to penalizing overly complex models, as in the Minimum Description Length (MDL) approach.

An exact Bayesian approach to model selection is usually unfeasible, since it involves computing the marginal likelihood $P(D) = \sum_G P(D, G)$, which is a sum over an exponential number of models (see Section 6.2). However, we can use an asymptotic approximation to the posterior called BIC (Bayesian Information Criterion), which is defined as follows:

$$\log \Pr(G|D) \approx \log \Pr(D|G, \hat{\Theta}_G) - \frac{\log N}{2} \#G$$

where N is the number of samples, $\#G$ is the dimension of the model[5], and $\hat{\Theta}_G$ is the ML estimate of the parameters.

[5]In the fully observable case, the dimension of a model is the number of free parameters. In a model with hidden variables, it might be less than this [GHM96].

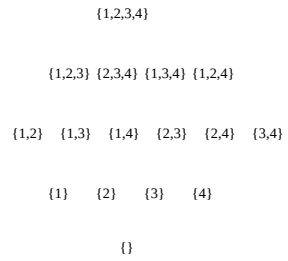

Figure 5: The subsets of $\{1, 2, 3, 4\}$ arranged in a lattice. The k'th row represents subsets of size k, for $0 \leq k \leq 4$.

The first term is computed using Equation 1. The second term is a penalty for model complexity. Since the number of parameters in the model is the sum of the number of parameters in each node[6], we see that the BIC score decomposes just like the log-likelihood (Equation 1). Hence all of the techniques invented for maximizing MI also work for the BIC case, and indeed, any decomposable scoring function.

6.2 Algorithms for finding the best model

Given that the score is decomposable, we can learn the parent set for each node independently. There are $\sum_{k=0}^{n} \binom{n}{k} = 2^n$ such sets, which can be arranged in a lattice as shown in Figure 5. The problem is to find the highest scoring point in this lattice.

The approach taken by REVEAL [LFS98] is to start at the bottom of the lattice, and evaluate the score at all points in each successive level, until a point is found with a score of 1.0. Since the scoring function they use is $I(X, \mathrm{Pa}(X)) / \max\{H(X), H(\mathrm{Pa}(X))\}$, where $I(X, Y)$ is the mutual information between X and Y, and $H(X)$ is the entropy of X (see Appendix A for definitions), 1.0 is the highest possible score, and corresponds to $\mathrm{Pa}(X)$ being a perfect predictor of X, i.e., $H(X|\mathrm{Pa}(X)) = 0$.

A normalized MI of 1 is only achievable with a deterministic relationship. For stochastic relationships, we must decide whether the gains in MI produced by a larger parent set is "worth it". The standard approach in the RA community uses the fact [Mil55] that $\chi^2(X, Y) \approx I(X, Y) N \ln(4)$, where N is the number of samples. Hence we can use a χ^2 test to decide whether an increase in MI is statistically significant. (This also gives us some kind of confidence measure on the connections that we learn.) Alternatively, we can use a complexity penalty term, as in the BIC score.

Of course, if we do not know if we have achieved the maximum possible score, we do not know when to stop searching, and hence we must evaluate all points in the lattice (although we can obviously use branch-and-bound). For large n, this is computationally infeasible, so a common approach

[6]Hence nodes with compact representations of their CPDs will encur a lower penalty, which can allow connections to form which might otherwise have been rejected [FG96].

is to only search up until level K (i.e., assume a bound on the maximum number of parents of each node), which takes $O(n^K)$ time. Unfortunately, in real genetic networks, it is known that some genes can have very high fan-in (and fan-out), so restricting the bound to $K = 3$ would make it impossible to discover these important "master" genes.

The obvious way to avoid the exponential cost (and the need for a bound, K) is to use heuristics to avoid examining all possible subsets. (In fact, we must use heuristics of some kind, since the problem of learning optimal structure is NP-hard [CHG95].) One approach in the RA framework, called Extended Dependency Analysis (EDA) [Con88], is as follows. Start by evaluating all subsets of size up to two, keep all the ones with significant (in the χ^2 sense) MI with the target node, and take the union of the resulting set as the set of parents.

The disadvantage of this greedy technique is that it will fail to find a set of parents unless some subset of size two has significant MI with the target variable. However, a Monte Carlo simulation in [Con88] shows that most random relations have this property. In addition, highly interdependent sets of parents (which might fail the pairwise MI test) violate the causal independence assumption, which is necessary to justify the use of noisy-OR and similar CPDs.

An alternative technique, popular in the UAI community, is to start with an initial guess of the model structure (i.e., at a specific point in the lattice), and then perform local search, i.e., evaluate the score of neighboring points in the lattice, and move to the best such point, until we reach a local optimum. We can use multiple restarts to try to find the global optimum, and to learn an ensemble of models. Note that, in the partially observable case, we need to have an initial guess of the model structure in order to estimate the values of the hidden nodes, and hence the (expected) score of each model (see Section 5); starting with the fully disconnected model (i.e., at the bottom of the lattice) would be a bad idea, since it would lead to a poor estimate.

6.3 Sample complexity

In addition to the computational cost, another important consideration is the amount of data that is required to reliably learn structure. For deterministic boolean networks, this issue has been addressed from a statistical physics perspective [Her98] and a computational learning theory (combinatorial) perspective [AMK99]. In particular, [AMK99] prove that, if the fan-in is bounded by a constant K, the number of samples needed to identify a boolean network of n nodes is lower bounded by $\Omega(2^K + K \log n)$ and upper bounded by $O(2^{2K} 2K \log n)$. Unfortunately, the constant factor is exponential in K, which can be quite large for certain genes, as we have already remarked. These analyses all assume a "tabula rasa" approach; however, in practice, we will often have strong prior knowledge about the structure (or at least parts of it), which can reduce the data requirements considerably.

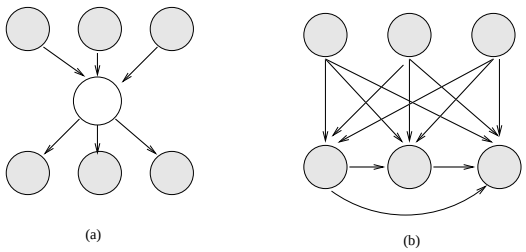

Figure 6: (a) A BN with a hidden variable H. (b) The simplest network that can capture the same distribution without using a hidden variable (created using arc reversal and node elimination). If H is binary and the other nodes are trinary, and we assume full CPTs, the first network has 45 independent parameters, and the second has 708.

6.4 Scaling up to large networks

Since there are about $n \approx 6400$ genes for yeast (S. Cerevisae), all of which can now be simultaneously measured [DLB97], it is clear that we will have to do some preprocessing to exclude the "irrelevant" genes, and just try to learn the connections between the rest. After all, many of them do not change their expression level during a given experiment. In the [DLB97] dataset, for example, which consists of 7 times steps, only 416 genes showed a significant change — the rest are completely uninformative. Of course, even $n = 416$ is too big for the techniques we have discussed, so we must pre-process further, perhaps by clustering genes with similar timeseries [ESBB98]. In Section 8.1, we discuss techniques for learning the structure of DBNs with continuous state, which scale much better than the methods we have discussed above for discrete models.

7 Unknown structure, partial observability

Finally, we come to the hardest case of all, where the structure is unknown and there are hidden variables and/or missing data. One difficulty is that the score does not decompose. However, as observed in [Fri97, Fri98, FMR98, TMCH98], the *expected* score *does* decompose, and so we can use an iterative method, which alternates between evaluating the expected score of a model (using an inference engine), and changing the model structure, until a local maximum is reached. This is called the Structural EM (SEM) algorithm. SEM was succesfully used to learn the structure of discrete DBNs within missing data in [FMR98].

7.1 Inventing new hidden nodes

So far, structure learning has meant finding the right connectivity between pre-existing nodes. A more interesting problem is inventing hidden nodes on demand. Hidden nodes can make a model much more compact: see Figure 6. The standard approach is to keep adding hidden nodes one at a time, to some part of the network (see below), performing structure learning at each step, until the score drops. One problem is choosing the cardinality (num-

ber of possible values) for the hidden node, and its type of CPD. Another problem is choosing where to add the new hidden node. There is no point making it a child, since hidden children can always be marginalized away, so we need to find an existing node which needs a new parent, when the current set of possible parents is not adequate.

[RM98] use the following heuristic for finding nodes which need new parents: they consider a noisy-OR node which is nearly always on, even if its non-leak parents are off, as an indicator that there is a missing parent. Generalizing this technique beyond noisy-ORs is an interesting open problem. One approach might be to examine $H(X|\text{Pa}(X))$: if this is very high, it means the current set of parents are inadequate to "explain" the residual entropy; if $\text{Pa}(X)$ is the best (in the BIC or χ^2 sense) set of parents we have been able to find in the current model, it suggests we need to create a new node and add it to $\text{Pa}(X)$.

A simple heuristic for inventing hidden nodes in the case of DBNs is to check if the Markov property is being violated for any particular node. If so, it suggests that we need connections to slices further back in time. Equivalently, we can add new lag variables (see Section 2.5) and connect to them. (Note that reasoning across multiple time slices forms the basis of the ("model free") correlational analysis approach of [AR95, ASR97].)

Another heuristic for DBNs might be to first perform a clustering of the timeseries of the observed variables (see e.g., [ESBB98]), and then to associate hidden nodes with each cluster. The result would be a Markov model with a tree-structured hidden "backbone" c.f., [JGS96]. This is one possible approach to the problem of learning hierarchically structured DBNs. (Building hierarchical (object oriented) BNs and DBNs *by hand* is straightforward, and there are algorithms which can exploit this modular structure to a certain extent to speed up inference [KP97, FKP98].)

Of course, interpreting the "meaning" of hidden nodes is always tricky, especially since they are often unidentifiable, e.g., we can often switch the interpretation of the true and false states (assuming for simplicity that the hidden node is binary) provided we also permute the parameters appropriately. (Symmetries such as this are one cause of the multiple maxima in the likelihood surface.) Our opinion is that fully automated structure discovery techniques can be useful as hypothesis generators, which can then be tested by experiment.

8 DBNs with continuous state

Up until now, we have assumed, for simplicity, that all the random variables are discrete-valued. However, in reality, most of the quantities we care about in genetic networks are continuous-valued. If we coarsely quantize the continuous variables (i.e., convert them to a small number of discrete values), we lose a lot of information, but if we finely quantize them, the resulting discrete model will have too many parameters. Hence it is better to work with continuous variables directly.

To create a BN with continuous variables, we need to specify the graph structure and the Conditional Probability Distributions at each node, as before. The most common distribution is a Gaussian, since it is analytically tractable. Consider, for example, the DBN in Figure 2(a), with $P(X_t = x_t|X_{t-1} = x_{t-1}) = N(x_t; \mu + W x_{t-1}, \mathbf{\Sigma})$, where

$$N(x; \mu, \mathbf{\Sigma}) = \frac{1}{Z} \exp\left((x-\mu)'|\mathbf{\Sigma}|^{-1}(x-\mu)\right) \quad (3)$$

is the Normal (Gaussian) distribution evaluated at x, $Z = (2\pi)^{n/2}|\mathbf{\Sigma}|^{\frac{1}{2}}$ is the normalizing constant to ensure the density integrates to 1, and W is the weight or regression matrix. (x' denotes the transpose of x.) Another way of writing this is $X_t = W X_{t-1} + \mu + \xi_t$, where $\xi_t \sim N(0, \mathbf{\Sigma})$ are independent Gaussian noise terms, corresponding to unmodelled influences. This is called a first-order auto-regressive AR(1) time series model [Ham94].

If X_t is a vector containing the expression levels of all the genes at time t, then this corresponds precisely to the model in [DWFS99]. (They do not explicitely mention Gaussian noise, but it is implicit in their decision to use least squares to fit the W term: see Section 8.1). Despite the simplicity of this model (in particular, its linear nature), they find that it models their experimental data very well.

We can also define nonlinear models. For example, if we define $X_t = g(W X_{t-1}) + \xi_t$, where $g(x) = 1/(1+\exp(-x))$ is the sigmoid function, we obtain the nonlinear model in [WWS99].

It is simple to extend both the linear and nonlinear models to allow for external inputs E_t, so that we regress X_t on X_{t-1} *and* E_t, as the authors point out. For example, in [DWFS99], they inject kainate — "a glutamatergic agonist which causes seizures, localized cell death, and severely disrupts the normal gene expression patterns" — into rat CNS, and measure expression levels before and after. Their model is therefore $X_t = W X_{t-1} + b K_t + \mu$, where $K_t = 0$ before the injection, and $K_t = \exp(-\tau/2)$ if t is τ hours after injection, and b and μ are vectors of length n, representing the response to kainate and an offset (bias) term, respectively.

In Figure 2(a), the X_t variables are always observed. Now consider Figure 2(b), where only the Y_t are observed, and have Gaussian CPDs. If the X_t are hidden, *discrete* variables, we have $P(Y_t = y|X_t = i) = N(y; \mu_i, \mathbf{\Sigma}_i)$; this is an HMM with Gaussian outputs. If the X_t are hidden, *continuous* variables, we have $P(Y_t = y|X_t = x) = N(y; \mu + Cx, \mathbf{\Sigma})$. This is called a Linear Dynamical System, since both the dynamics, $P(X_t|X_{t-1})$, and the observations, $P(Y_t|X_t)$, are linear. (It is also called a State Space model.)

To perform inference in an LDS, we can use the well-known Kalman filter algorithm [BSL93], which is just a special case of the junction tree algorithm. If the dynamics are non-linear, however, inference becomes much harder; the standard technique is to make a local linear approximation — this is called the Extended Kalman Filter (EKF), and is similar to techniques developed in the Recurrent

Neural Networks literature [Pea95]. (One reason for the widespread success of HMMs with Gaussian outputs is that the discrete hidden variables can be used to approximate arbitrary non-linear dynamics, given enough training data.) In the rest of this paper, we will stick to the case of fully observable (but possibly non-linear) models, for simplicity, so that inference will not be necessary. (We will, however, consider Bayesian methods of parameter learning, which can be thought of as inferring the most probable values of nodes which represent the parameters themselves.)

8.1 Learning

We have assumed that X_t is a *vector* of random variables. If we were to represent each component of this vector as its own (scalar) node, the inter-slice connections would correspond to the non-zero values in the weight matrix W, and *undirected* connections *within* a slice would correspond to non-zero entries in Σ^{-1} [Lau96]. For example, if $W = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix}$, and Σ is diagonal, the model is equivalent to the one in Figure 3(a). Hence, in the case of continuous state models, we can convert back and forth between representing the structure graphically and numerically. This means that structure learning reduces to parameter fitting, and we can avoid expensive search through a discrete space of models! (c.f., [Bra98].) Consequently, we will now discuss techniques for learning the parameters of continuous-state DBNs.

Consider the AR(1) model in Figure 2(a). For notational simplicity we will assume $\Sigma = \sigma^2 I$, so the normalizing constant is $Z = (\frac{2\pi}{\beta})^{n/2}$, where $\beta = 1/\sigma^2$ is the inverse variance (the precision). Using Equation 3, we find that the likelihood of the data (ignoring terms which are independent of W) is

$$\begin{aligned} P(D|W) &= \prod_{l=1}^{S} \prod_{t=2}^{T} P(X_t^l | X_{t-1}^l) \\ &= \frac{1}{Z_D(\beta)} \exp\left(\sum_{l,t} -\frac{\beta}{2} ||X_t^l - W X_{t-1}^l||^2 \right) \\ &= \frac{1}{Z_D(\beta)} \exp(-\beta E_D) \end{aligned} \quad (4)$$

where X_t^l is the value of X at time t in sequence l, $Z_D(\beta) = Z^N$ is the product of the individual normalizing constants, $N = S(T-1)$ is the number of samples, and E_D is an error (cost) function on the data. (Having multiple sequences is essentially the same as concatenating them into a single long sequence (modulo boundary conditions), so we shall henceforth just use a single index, t.) Differentiating this with respect to W and setting to 0 yields

$$W = \left(\sum_{t=2}^{T} X_t X_{t-1} \right) \left(\sum_{t=2}^{T} X_t X_t \right)^{-1}$$

Let us now rewrite this in a more familiar form. Define the matrices

$$X = \begin{pmatrix} X_1' \\ \vdots \\ X_{T-1}' \end{pmatrix}, \quad Y = \begin{pmatrix} X_2' \\ \vdots \\ X_T' \end{pmatrix}$$

We can think of the t'th row of X as the t'th input vector to a linear neural network with no hidden layers (i.e., a perceptron), and the t'th row of Y as the corresponding output. Using this notation, we can rewrite the above expression as follows (this is called the normal equation):

$$W' = (X'X)^{-1} X'Y = X^\dagger Y$$

where $X^\dagger$ is the pseudo-inverse of X (which is not square). Thus we see that the least squares solution is the same as the Maximum Likelihood (ML) solution assuming Gaussian noise.

The technique used by D'haeseleer et al. [DWFS99] is to compute the least squares value of W, and interpret $|W_{i,j}| < \theta$, for some (unspecified) threshold θ, as indicating the absence of an arc between nodes i and j. However, as Bishop writes [Bis95, p.360], "[this technique] has little theoretical motivation, and performs poorly in practice".

More sophisticated techniques have been developed in the neural network literature, which involve examining the change in the error function due to small changes in the values of the weights. This requires computing the Hessian H of the error surface. In the technique of "optimal brain damage" [CDS90], they assume H is diagonal; in the more sophisticated technique of "optimal brain surgeon" [HS93], they remove this assumption.

Since we believe (hope!) that the true model is sparse (i.e., most entries in W are 0), we can encode this knowledge (assumption) by using a $N(0, 1/\alpha)$ Gaussian prior[7] for each weight:

$$\begin{aligned} P(W) &= \prod_{ij} P(W_{ij}) = \prod_{ij} \frac{1}{(2\pi/\alpha)^{\frac{1}{2}}} \exp\left(\frac{-\alpha}{2} W_{ij}^2 \right) \\ &= \frac{1}{Z_W(\alpha)} \exp\left(\frac{-\alpha}{2} \sum_{ij} W_{ij}^2 \right) \\ &= \frac{1}{Z_W(\alpha)} \exp(-\alpha E_W) \end{aligned} \quad (5)$$

where $Z_W(\alpha) = (\frac{2\pi}{\alpha})^{n^2/2}$ (since W is an $n \times n$ matrix) and E_W is an error (cost) function on the weights. Then, using Bayes' rule (Equation 2) and Equations 4 and 5, the posterior distribution on the weights is

$$P(W|D) = \frac{1}{Z_S(\alpha, \beta)} \exp(-\beta E_D - \alpha E_W)$$

Since the denominator is independent of W, the Maximum Posterior (MAP) value of W is given by minimizing the

[7] In [WWS99], they use an ad-hoc iterative pruning method to achieve a similar effect.

cost function

$$S(W) = \beta E_D + \alpha E_W = \beta E_D + \alpha \frac{1}{2} \sum_{i,j} W_{i,j}^2$$

The second term is called a regularizer, and encourages learning of a weight matrix with small values. (Unfortunately, this prior favours many small weights, rather than a few large ones, although this can be fixed [HP89].) This regularization technique is also called weight decay. Note that the use of a regularizer overcomes the problem that W will often be underdetermined, i.e., there will be fewer samples than parameters, $Nn < n^2$.

The values α and β are called hyperparameters, since they control the prior on the weights and the system noise, respectively. Since their values are unknown, the correct Bayesian approach is to integrate them out, but an approximation that actually works better in practice is to approximate their posterior by a Gaussian (or maybe a mixture of Gaussians), find their MAP values, and then plug them in to the above equation: see [Bis95, sec. 10.4] and [Mac95] for details.

We can associate a separate hyperparameter $\alpha_{i,j}$ for each weight $W_{i,j}$, find their MAP values, and use this as a "soft" means of finding which entries of $W_{i,j}$ to keep: this is called Automatic Relevance Determination (ARD) [Mac95], and has been found to peform better than the weight deletion techniques discussed above, especially on small data sets.

9 Conclusion

In this paper, we have explained what DBNs are, and discussed how to learn them. In the future, we hope to apply these techniques (in particular, the ones discussed in Section 8.1) to real biological data, when enough becomes publicly available.[8]

A The equivalence between Mutual Information and Maximum Likelihood methods

We start by introducing some standard notation from information theory [CT91].

$$\begin{aligned} I(X,Y) &= \sum_{x,y} P(x,y) \log \frac{P(x,y)}{P(x)P(y)} \\ &= H(X) - H(X|Y) \\ &= H(X) + H(Y) - H(X,Y) \end{aligned}$$

is the mutual information (MI) between X and Y,

$$H(X) = -\sum_x P(x) \log P(x)$$

is the entropy of X, and

$$H(X|Y) = -\sum_{x,y} P(y)P(x|y) \log P(x|y)$$

is the conditional entropy of X given Y. (If $\text{Pa}(X_i) = \emptyset$, we define $H(X_i|\text{Pa}(X_i)) = H(X_i)$ and $I(X_i, \text{Pa}(X_i)) = 0$.) Finally, we define the MI score of a model as $MIS(G;D) = \sum_i I(X_i, \text{Pa}_G(X_i); D)$, where the probabilities are set to their empirical values given D, and $\text{Pa}_G(X_i)$ mean the parents of node X_i in graph G.

Theorem. Let G, G' be BNs in which every node has a full CPT, and D be a fully observable data set. Then[9]

$$L(G;D) - L(G';D) = MIS(G;D) - MIS(G';D)$$

Proof: To see this, note that the normalized log-likelihood (Equation 1) can be rewritten as

$$L = \frac{1}{N} \sum_{i,j,k} N_{ijk} \log \Pr(X_i = k | \pi_i = j)$$

where we have defined $\pi_i \stackrel{\text{def}}{=} \text{Pa}(X_i)$ for brevity. N_{ijk} is the number of times the event $(X_i = k, \pi_i = j)$ occurs in the whole training set. (If $\pi_i = \emptyset$, we define $\Pr(X_i = k|\pi_i = j) = \Pr(X_i = k)$ and N_{ijk} as the number of times the event $X_i = k$ occurs.) If we assume each node is a full CPT, and set the probabilities to their empirical estimates, then $N_{ijk} = N \Pr(X_i = k, \pi_i = j)$, and so

$$\begin{aligned} L = & \sum_{i,j,k} \Pr(X_i = k, \pi_i = j) \log \Pr(X_i = k | \pi_i = j) \\ = & \sum_i -H(X_i|\pi_i) \\ = & \sum_i I(X_i, \pi_i) - H(X_i) \end{aligned}$$

Since $H(X_i)$ is independent of the structure of the graph G, we find $L(G;D) - L(G';D) = \sum_i I(X_i, \text{Pa}_G(X_i)) - I(X_i, \text{Pa}_{G'}(X_i))$. ∎

We note that this theorem is similar to the result in [CL68], who show that the optimal [10] tree-structured MRF is a maximal weight spanning tree (MWST) of the graph in which the weight of an arc between two nodes, X_i and X_j, is $I(X_i, X_j)$. [MJM98] extends this result to mixtures of trees. Trees have the significant advantage that we can

[8] We carried out some provisional experiments on a subset of the data described in [WFM+98] (available at `rsb.info.nih.gov/mol-physiol/PNAS/GEMtable.html`), which has $n = 112$, $T = 9$ and $S = 1$. Applying the ARD technique to the 17 genes involved in the GAD/GABA subsystem, described in [DWFS99], we found that all the $\alpha_{i,j}$'s were significant, indicating a fully connected graph. [DWFS99] got a sparse graph, but used a larger (unpublished) data set with $T = 28$, and used an unspecified pruning threshold θ.

[9] If $f(G) - f(G') = g(G) - g(G')$, then $f(G) > f(G') \iff g(G) > g(G')$, so f and g rank models in the same order. However, there might be many models which receive the same score under f or g, so it is ambiguous to say $\arg\max_G f(G) = \arg\max_G g(G)$.

[10] In the sense of minimizing the KL divergence between the true and estimated distribution.

compute the MWST in $O(n^2)$ time, where n is the number of variables. However, it is not clear if they are a useful model for gene expression data.

If a node does not have a full CPT, but instead has a parametric CPD *of the right form* (i.e., at least capable of representing the "true" conditional probability distribution of the node), then as $N \to \infty$, $N_{ijk} \to N \Pr(X_i = k|\pi_j)$, and the proof goes through as before. But this leaves us with the additional problem of choosing the form of CPD: see Section 4.2.

References

[AMK99] T. Akutsu, S. Miyano, and S. Kuhara. Identification of genetic networks from a small number of gene expression patterns under the boolean network model. In *Proc. of the Pacific Symp. on Biocomputing*, 1999.

[AR95] A. Arkin and J. Ross. Statistical construction of chemical reaction mechanisms from measured time-series. *J. Phys. Chem.*, 99:970, 1995.

[ASR97] A. Arkin, P. Shen, and J. Ross. A test case of correlation metric construction of a reaction pathway from measurements. *Science*, 277:1275–1279, 1997.

[BFGK96] C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in bayesian networks. In *Proc. of the Conf. on Uncertainty in AI*, 1996.

[Bis95] C. M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, 1995.

[BK98a] X. Boyen and D. Koller. Approximate learning of dynamic models. In *Neural Info. Proc. Systems*, 1998.

[BK98b] X. Boyen and D. Koller. Tractable inference for complex stochastic processes. In *Proc. of the Conf. on Uncertainty in AI*, 1998.

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

[Bra98] M. Brand. An entropic estimator for structure discovery. In *Neural Info. Proc. Systems*, 1998.

[BSL93] Y. Bar-Shalom and X. Li. *Estimation and Tracking: Principles, Techniques and Software*. Artech House, 1993.

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

[Bun96] W. Buntine. A guide to the literature on learning probabilistic networks from data. *IEEE Trans. on Knowledge and Data Engineering*, 8(2), 1996.

[CDS90] Y. Le Cun, J. Denker, and S. Solla. Optimal brain damage. In *Neural Info. Proc. Systems*, 1990.

[CHG95] D. Chickering, D. Heckerman, and D. Geiger. Learning Bayesian netwprks: search methods and experimental results. In *AI + Stats*, 1995.

[CL68] C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Trans. on Info. Theory*, 14:462–67, 1968.

[Con88] R. C. Conant. Extended dependency analysis of large systems. *Intl. J. General Systems*, 14:97–141, 1988.

[CT91] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley, 1991.

[Dec98] R. Dechter. Bucket elimination: a unifying framework for probabilistic inference. In M. Jordan, editor, *Learning in Graphical Models*. Kluwer, 1998.

[DEKM98] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, Cambridge, 1998.

[DLB97] J. L. DeRisi, V. R. Lyer, and P. O. Brown. Exploring the metabolic and genetic control of gene expression on a genomic scale. *Science*, 278:680–686, 1997.

[DWFS99] P. D'haeseleer, X. Wen, S. Fuhrman, and R. Somogyi. Linear modeling of mRNA expression levels during CNS development and injury. In *Proc. of the Pacific Symp. on Biocomputing*, 1999.

[ESBB98] Michael B. Eisen, Paul T. Spellman, Patrick O. Brown, and David Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc. of the National Academy of Science, USA*, 1998.

[FG96] N. Friedman and M. Goldszmidt. Learning Bayesian networks with local structure. In *Proc. of the Conf. on Uncertainty in AI*, 1996.

[FKP98] N. Friedman, D. Koller, and A. Pfeffer. Structured representation of complex stochastic systems. In *Proc. of the Conf. of the Am. Assoc. for AI*, 1998.

[FMR98] N. Friedman, K. Murphy, and S. Russell. Learning the structure of dynamic probabilistic networks. In *Proc. of the Conf. on Uncertainty in AI*, 1998.

[Fre97] D. A. Freeman. From association to causation via regression. In V. R. McKim and S. P. Turner, editors, *Causality in Crisis? Statistical methods and the search for causal knowledge in the social sciences*. U. Notre Dame Press, 1997.

[Fri97] N. Friedman. Learning Bayesian networks in the presence of missing values and hidden variables. In *Proc. of the Conf. on Uncertainty in AI*, 1997.

[Fri98] N. Friedman. The bayesian structural EM algorithm. In *Proc. of the Conf. on Uncertainty in AI*, 1998.

[GHM96] D. Geiger, D. Heckerman, and C. Meek. Asymptotic model selection for directed networks with hidden variables. In *Proc. of the Conf. on Uncertainty in AI*, 1996.

[Gol80] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, 1980.

[Ham94] J. Hamilton. *Time Series Analysis*. Wiley, 1994.

[HB94] D. Heckerman and J.S. Breese. Causal independence for probability assessment and inference using bayesian networks. *IEEE Trans. on Systems, Man and Cybernetics*, 26(6):826–831, 1994.

[HD94] C. Huang and A. Darwiche. Inference in belief networks: A procedural guide. *Intl. J. Approx. Reasoning*, 11, 1994.

[Hec96] D. Heckerman. A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, March 1996.

[Hen89] M. Henrion. Some practical issues in constructing belief networks. In *Proc. of the Conf. on Uncertainty in AI*, 1989.

[Her98] J. Hertz. Statistical issues in the reverse engineering of genetic networks. In *Proc. of the Pacific Symp. on Biocomputing*, 1998.

[HMC97] D. Heckerman, C. Meek, and G. Cooper. A Bayesian approach to causal discovery. Technical Report MSR-TR-97-05, Microsoft Research, February 1997.

[HP89] S. Hanson and L. Pratt. Comparing biases for minimal network construction with back-propogation. In *Neural Info. Proc. Systems*, 1989.

[HS93] B. Hassibi and D. Stork. Second order derivatives for network pruning: optimal brain surgeon. In *Neural Info. Proc. Systems*, 1993.

[HS95] D. Heckerman and R. Shachter. Decision-theoretic foundations for causal reasoning. *J. of AI Research*, 3:405–430, 1995.

[Jen96] F. V. Jensen. *An Introduction to Bayesian Networks*. UCL Press, London, England, 1996.

[JGJS98] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. In M. Jordan, editor, *Learning in Graphical Models*. Kluwer, 1998.

[JGS96] M. I. Jordan, Z. Ghahramani, and L. K. Saul. Hidden Markov decision trees. In *Neural Info. Proc. Systems*, 1996.

[Kau93] S. Kauffman. *The origins of order. Self-organization and selection in evolution*. Oxford Univ. Press, 1993.

[Kli85] G. Klir. *The Architecture of Systems Problem Solving*. Plenum Press, 1985.

[KP97] D. Koller and A. Pfeffer. Object-oriented bayesian networks. In *Proc. of the Conf. on Uncertainty in AI*, 1997.

[Kri86] K. Krippendorff. *Information Theory : Structural Models for Qualitative Data (Quantitative Applications in the Social Sciences, No 62)*. Page Publishers, 1986.

[Lau95] S. L. Lauritzen. The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, 19:191–201, 1995.

[Lau96] S. Lauritzen. *Graphical Models*. OUP, 1996.

[LFS98] S. Liang, S. Fuhrman, and R. Somogyi. Reveal, a general reverse engineering algorithm for inference of genetic network architectures. In *Pacific Symposium on Biocomputing*, volume 3, pages 18–29, 1998.

[MA97] H. H. McAdams and A. Arkin. Stochastic mechanisms in gene expression. *Proc. of the National Academy of Science, USA*, 94(3):814–819, 1997.

[Mac95] D. MacKay. Probable networks and plausible predictions — a review of practical Bayesian methods for supervised neural networks. *Network*, 1995.

[Mac98] D. MacKay. Introduction to monte carlo methods. In M. Jordan, editor, *Learning in graphical models*. Kluwer, 1998.

[MH97] C. Meek and D. Heckerman. Structure and parameter learning for causal independence and causal interaction models. In *Proc. of the Conf. on Uncertainty in AI*, pages 366–375, 1997.

[Mil55] G. Miller. Note on the bias of information estimates. In H. Quastler, editor, *Information theory in psychology*. The Free Press, 1955.

[MJM98] M. Meila, M. Jordan, and Q. Morris. Estimating dependency structure as a hidden variable. Technical Report 1648, MIT AI Lab, 1998.

[Mur99] K. P. Murphy. A variational approximation for Bayesian networks with discrete and continuous latent variables. In *Proc. of the Conf. on Uncertainty in AI*, 1999. Submitted.

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

[Pea95] B. A. Pearlmutter. Gradient calculations for dynamic recurrent neural networks: a survey. *IEEE Trans. on Neural Networks*, 6, 1995.

[Rab89] L. R. Rabiner. A tutorial in Hidden Markov Models and selected applications in speech recognition. *Proc. of the IEEE*, 77(2):257–286, 1989.

[RM98] S. Ramachandran and R. Mooney. Theory refinement of bayesian networks with hidden variables. In *Machine Learning: Proceedings of the International Conference*, 1998.

[SGS93] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. Springer-Verlag, 1993. Out of print.

[Sha98] R. Shachter. Bayes-ball: The rational pastime (for determining irrelevance and requisite information in belief networks and influence diagrams). In *Proc. of the Conf. on Uncertainty in AI*, 1998.

[SHJ97] P. Smyth, D. Heckerman, and M. I. Jordan. Probabilistic independence networks for hidden Markov probability models. *Neural Computation*, 9(2):227–269, 1997.

[Sri93] S. Srinivas. A generalization of the noisy-or model. In *Proc. of the Conf. on Uncertainty in AI*, 1993.

[SS96] R. Somogyi and C. A. Sniegoski. Modeling the complexity of genetic networks: understanding multigenetic and pleiotropic regulation. *Complexity*, 1(45), 1996.

[TMCH98] B. Thiesson, C. Meek, D. Chickering, and D. Heckerman. Learning mixtures of DAG models. In *Proc. of the Conf. on Uncertainty in AI*, 1998.

[WFM+98] X. Wen, S. Fuhrman, G. S. Michaels, D. B. Carr, S. Smith, J. L. Barker, and R. Somogyi. Large-scale temporal gene expression mapping of central nervous system development. *Proc. of the National Academy of Science, USA*, 95:334–339, 1998.

[Whi90] J. Whittaker. *Graphical Models in Applied Multivariate Statistics*. Wiley, 1990.

[WMS94] J. V. White, I. Muchnik, and T. F. Smith. Modelling protein cores with Markov Random Fields. *Mathematical Biosciences*, 124:149–179, 1994.

[WWS99] D. C. Weaver, C. T. Workman, and G. D. Stormo. Modeling regulatory networks with weight matrices. In *Proc. of the Pacific Symp. on Biocomputing*, 1999.