EXPERIMENTAL DESIGN OF TIME SERIES DATA FOR LEARNING FROM DYNAMIC BAYESIAN NETWORKS

DAVID PAGE AND IRENE M. ONG

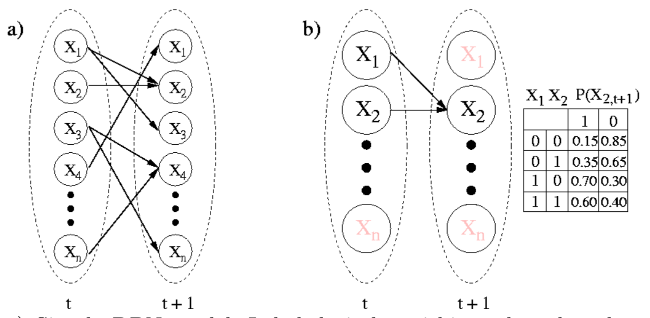
Department of Biostatistics & Medical Informatics
University of Wisconsin
Madison, WI 53706 USA
E-mail: page@biostat.wisc.edu, ong@cs.wisc.edu

Bayesian networks (BNs) and dynamic Bayesian networks (DBNs) are becoming more widely used as a way to learn various types of networks, including cellular signaling networks[16,14], from high-throughput data. Due to the high cost of performing experiments, we are interested in developing an experimental design for time series data generation. Specifically, we are interested in determining properties of time series data that make them more efficient for DBN modeling. We present a theoretical analysis on the ability of DBNs without hidden variables to learn from proteomic time series data. The analysis reveals, among other lessons, that under a reasonable set of assumptions a fixed budget is better spent on collecting many short time series data than on a few long time series data.

1. Introduction

Time series data, and dynamic Bayesian networks (DBNs) to model such data, have become more popular as an approach to learning gene regulatory networks[3,12,6,13]. As experimental techniques for collecting proteomic data has improved, there has been a shift to learning protein networks from various types of proteomic data[16,20]. However, the cost of generating experimental proteomic data is still high, hence, we are interested in developing an experimental design for time series data generation that is more efficient for dynamic Bayesian network learning.

In a typical experiment, mass spectrometry (MS) is used to measure protein abundance *at several specific time points* after a particular stimulus to an organism or cell sample. The peaks in the MS data can be discretized to indicate the presence or absence of protein, avoiding certain problems associated with continuous values (S. McIlwain, personal communication 2005). From the discretized data, we can then learn a DBN model that best fits the data. Biologists can visually examine such a DBN structure (Fig. 1a) and interpret an arc from protein X_1 at time t to protein X_2 at



X_1	X_2	$P(X_{2,t+1})$ 1	0
0	0	0.15	0.85
0	1	0.35	0.65
1	0	0.70	0.30
1	1	0.60	0.40

Figure 1. a) Simple DBN model. Labeled circles within a dotted oval represent our variables in one time slice. Formally, arcs connecting variables from one time slice to variables in the next have the same meaning as in a BN, but they intuitively carry a stronger implication of causality. b) Example of probabilistic CPTs.

time $t + 1$ as evidence that X_1 influences X_2.

Following earlier experiments in learning DBNs from time series data[13], we consulted with biologists about the design of future time series experiments. While a number of design issues arise, the most common question is the following. "Given that we have resources to run r experiments, is it better to run many short time series or a few long time series?" Design issues also arise for our learning algorithms. For example, given a specific number of experiments r that will be run, and a given amount of time to learn a DBN model from the data, should we place a limit on the number of parents a node can have and, if so, what should this limit be? One way to answer these questions is to perform many runs with many time series data sets having different properties; unfortunately, few such data sets are available, and the cost of producing such a data set requires design insight now, before additional data sets are available. An alternative way to gain insight is to construct a formal model of the learning task, as realistic as possible though necessarily making some simplifying assumptions.

The present paper limits its attention to DBNs whose variables are Boolean, though the results extend naturally to non-Boolean discrete variables. Because of the use of Boolean variables, our DBNs also can be viewed as (deterministic or probabilistic) Boolean networks. Friedman and Yakhini[5] have examined the sample complexity of minimum description length based learning procedures for Bayesian network structures and Dasgupta[4] determined the sample complexity of learning fixed-structure Bayesian networks with and without hidden nodes using the PAC (probably approximately correct) framework. Furthermore, Akutsu *et al.*[1] formalized the task of constructing Boolean networks from data and others extended or improved their initial results[2,17,10]. Nevertheless, the results do not give guarantees about the accuracy of the learned network on new or unseen data, or the amount of data required to achieve a given level of accuracy. The present paper addresses the question of polynomial-time learnability using the PAC-learning framework[18] and its extension to probabilistic concepts[8]. Consequently, the novel aspect of this paper is its provision of (probabilistic) accuracy guarantees based on data set size. The formal results in this paper, imply the following practical advice for the design of time series MS experiments that will be analyzed using DBNs.

First, many short time series experiments, specifically two time steps, are preferable to a single long time series experiment.

Second, given the number of time series experiments that can feasibly be run with present-day costs, the number of parents per node in a DBN should be limited to at most three per node.

Third, even with only two parents per node, the worst-case number of examples required to guarantee a given level of accuracy with a given probability is cubic in the number of variables n, and this number typically is in the thousands. If we are concerned with gaining insight into—or accurate prediction of—only a small number m of the n variables, we can reduce this term to n^2m. This often is the case where we are interested in learning or refining a model of a particular pathway, and we know most of the key players (proteins). If in addition we have an over-estimate of the potential other players, and there are l of these, then we can reduce this term further to l^2m, reducing the number of required experimental data.

2. Definitions and Terminology

The PAC framework allows us to provide upper and lower bounds on the number of examples needed to learn a hypothesis class assuming that the learning algorithm is making use of all the data. We formulate our definitions for the PAC framework below in the style of Kearns and Vazirani[9].

Definition 2.1. A Boolean *dynamic Bayesian network* (DBN) is defined over the Boolean variables

$$\begin{array}{l} X_{1,1}, X_{2,1}, \ldots, X_{n,1}, \\ X_{1,2}, X_{2,2}, \ldots, X_{n,2}, \\ \ldots, \\ X_{1,T}, X_{2,T}, \ldots, X_{n,T} \end{array}$$

where $X_{i,t}$ denotes variable X_i at time t. For each $1 \leq i \leq n$ and $1 < t \leq T$ the value of variable $X_{i,t}$ is $f_i(X_{1,t-1}, \ldots, X_{n,t-1})$, where f_i is some (possibly stochastic) Boolean function.

Definition 2.2. We denote by $\mathcal{DBN}(\mathcal{C}_n)$ the class of Boolean DBNs for which each function f_i comes from Boolean concept class $\mathcal{C}_n$.

Any particular Boolean DBN in $\mathcal{DBN}(\mathcal{C}_n)$ is a set of functions $f_i(X_{1,t-1}, \ldots, X_{n,t-1})$, one for each variable X_i, $1 \le i \le n$ (f_i does not change with time). For example, the Boolean concept class $\mathcal{C}_n$ might be all stochastic functions of at most k variables. This class of functions corresponds to all possible conditional probability tables (CPTs) in a DBN for a node with at most k parents. An example of such a CPT is given in Fig. 1b. If the DBN is deterministic, $\mathcal{C}_n$ might be the set of all (non-stochastic) functions of at most k variables, that is, all truth tables over k variables. For such a CPT, each row in Fig. 1b would instead have one of the probabilities set to 1 and the other set to 0. A generalization of this class, allowing more than k parents in a still limited fashion would be to have as $\mathcal{C}_n$ the set of all functions that can be represented by a k disjunctive normal form (kDNF).

3. Results

3.1. *Boolean DBN from 2-slice data*

Before presenting the first of our related models, we establish some conventions. In practice a DBN model may contain some variables that cannot be observed or measured; such variables are known as hidden variables. The present paper does not consider hidden variables or missing data.

In generating a MS data set, the cost of r experiments (measuring the abundance of each protein in r samples) is the same regardless of whether the r samples are all part of a single, long time series or many different time series. Therefore, we treat our number of data points as the number of experiments rather than the number of time series.

In the ordinary PAC-learning model, one assumes each data point is drawn randomly, independently according to some probability distribution D. Our models cannot assume this, because in a time series each data point (after the first) depends on the previous data point. The most faithful we can remain to the original PAC-learning model is to specify that the first data point in each time series is drawn randomly according to some probability distribution D, and the first data points in different time series are drawn independently of one another.

For simplicity, we begin with a formal model of DBN learning that resembles the PAC-learning model as much as possible, by restricting consideration to deterministic concepts. Given a deterministic DBN and a specific (input) time slice, the next (output) time slice is fixed according to the DBN. We say that a DBN model and a target DBN disagree with one another on an input time slice iff, given the input time slice, the two DBNs produce different outputs. A DBN model is $(1-\epsilon)$-accurate with respect to a target model iff the sum of the probabilities, according to D, of input time slices on which the two DBNs disagree is at most ϵ. As is standard with the PAC model, we take $|T|$ to denote the size of the target concept (DBN model) $T \in \mathcal{DBN}(\mathcal{C}_n)$ in a "reasonable" encoding scheme. For concreteness, we specify $|T|$ as the number of bits required to encode, for each variable X_i, its parents and its function f_i. The following definition is an application of the PAC-learning model to DBNs.

Definition 3.1. An algorithm PAC-learns a deterministic $\mathcal{DBN}(\mathcal{C}_n)$ iff there exist polynomials $\text{poly}_1(\_,\_,\_,\_)$ and $\text{poly}_2(\_)$ such that for any target DBN T in $\mathcal{DBN}(\mathcal{C}_n)$, any $0 < \epsilon < 1$ and $0 < \delta < 1$, and any probability distribution D over initial data points for time series: given any $r \geq \text{poly}_1(n,|T|,\frac{1}{\epsilon},\frac{1}{\delta})$ data points, the algorithm runs in $\text{poly}_2(rn)$ and with probability at least $1-\delta$ outputs a model that is $(1-\epsilon)$-accurate wrt T.

Theorem 3.1. *For any fixed $k \in \mathbb{N}$ the class of $\mathcal{DBN}$(kDNF) is PAC-learnable from 2-slice data.*

Proof. Algorithm A learns a kDNF formula to predict each of the n variables at time slice 2 from the values of the n variables at time slice 1. Each 2-slice time series (input and output) is used to generate one example for each $X_{i,2}$. For each $1 \leq i \leq n$ the output (class) is $X_{i,2}$ and input features are $X_{1,1}, \ldots, X_{n,1}$. Given a PAC learning algorithm L for kDNF expressions[7], we run L on n feature vectors to find a concept in $\mathcal{C}_n$.

Algorithm A iterates: for each variable $X_{i,2}$, $1 \leq i \leq n$, we call kDNF learning algorithm L with $\frac{\delta}{n}$ as the maximum probability of failure (i.e., with desired confidence of $1-\frac{\delta}{n}$) and with $\frac{\epsilon}{n}$ as the maximum error (i.e., with desired accuracy of $1-\frac{\epsilon}{n}$). Algorithm A's final model is the set of functions $f_i(X_{1,1},\ldots,X_{n,1})$ returned by L, one per output variable $X_{i,2}$.

Algorithm A runs in polynomial time since $n*\text{poly}_1(n,|T|,\frac{n}{\epsilon},\frac{n}{\delta})$ yields a polynomial, and each call to L runs in time polynomial in the size of its input. It remains only to show that with probability $1-\delta$ the error is bounded by ϵ. The definition of union bound states that if A and B are any two events (i.e., subsets of a probability space), then $\Pr(A \cup B) \leq \Pr(A)+\Pr(B)$[9]. Since each call to L fails to achieve the desired accuracy with probability only $\frac{\delta}{n}$, by the union bound the probability that there exists *any* of the n calls to L that fails to achieve the desired accuracy is at most δ. If each call to L has a desired error bound of $\frac{\epsilon}{n}$, then the error of the model (probability according to D of drawing an input time slice on which the learned model and target will disagree for some variable $X_{i,2}$, $1 \leq i \leq n$) is the union of all n error expressions from L, i.e., $\Pr(\mathit{Error}) = \frac{\epsilon}{n} + \frac{\epsilon}{n} + \ldots + \frac{\epsilon}{n} \leq \epsilon$. □

kDNF is a richer representation than one usually uses in a DBN. Typically, each variable is a function (represented as a CPT) of up to k parents. We denote the class of such DBNs by $\mathcal{DBN}$(k-parents). While PAC-learnability of a more restricted class does not automatically follow from PAC-learnability of a more general class, in this case arguments very similar to those just given show that, for any fixed $k \in \mathbb{N}$, the class of deterministic $\mathcal{DBN}$(k-parents) is PAC-learnable from 2-slice data.

3.2. *Boolean DBN from r-slice data*

It is equally common in practice for time series measurements to yield one long time series instead of multiple time series of length 2, or somewhere between these two extremes. While the *total number of experiments* r is determined largely by budget, the choice of time series *lengths* for any *fixed total* number of MS experiments r usually is not driven by expense. Rather, researchers make the choice they believe will provide the most information, because r experiments will have the same cost regardless of whether they occur in one long time series or many shorter time series.

We now ask whether the class $\mathcal{DBN}$(k-parents) is PAC-learnable from a single time series, and if so, whether the total number of experiments required might be less. It is trivial to prove that no algorithm PAC-learns this class when all the data points are in a single time series; the algorithm simply cannot learn enough about the distribution D according to which the start of each time series is drawn. But such a trivial negative result in unsatisfying. In practice if we subject a cell to an experimental condition and run a long time series of measurements, it is because we wish to learn an accurate model of how the organism responds to *that particular condition.* Therefore, we next consider a natural variant of our first learning model, where this variant is tailored to data points in a single time series.

Definition 3.2. An algorithm learns a deterministic class $\mathcal{DBN}(\mathcal{C}_n)$ from a *single time series* iff there exists polynomials $\text{poly}_1(\_,\_,\_,\_)$ and $\text{poly}_2(\_)$ such that for any target DBN T in $\mathcal{DBN}(\mathcal{C}_n)$, any $0 < \epsilon < 1$ and $0 < \delta < 1$, and any starting point for the time series: given a time series of any length $r \geq \text{poly}_1(n,|T|,\frac{1}{\epsilon},\frac{1}{\delta})$, the algorithm runs in $\text{poly}_2(rn)$ and with probability at least $1-\delta$ outputs a model that with probability at least $(1-\epsilon)$ correctly predicts time slice $r+1$.

Notice that we do not require that the learning algorithm be capable of performing well for most starting points, but only for the one given. For deterministic DBN models, after some m time slices the time series must return to a previous state, from which point on the time series will cycle with some period length at most m. If for some class of DBN models m is only polynomial in the number of variables n then it will be possible to PAC-learn this class of models from a single time series, within the definition just given. Unfortunately, even for the simple class of deterministic $\mathcal{DBN}(k$-parents), the period is superpolynomial in n and the size of the target model, leading to the following negative result.

Theorem 3.2. *For any $k \geq 2$ the class of $\mathcal{DBN}$ (k-parents) is not learnable from a single time series.*

Proof. Assume there exists a learning algorithm L for $\mathcal{DBN}(k$-parents). Then for any k-parent target DBN T, any $0 < \epsilon < 1$ and any $0 < \delta < 1$, given a time series of any length $r \geq \text{poly}_1(n,|T|,\frac{1}{\epsilon},\frac{1}{\delta})$, L will run in time polynomial in the size of its input and with probability at least $1 - \delta$ will output a model that will correctly predict time slice $r+1$ with probability at least $1-\epsilon$. Because any 2-parent DBN can be represented in a number of bits that is polynomial in n, we can simplify $\text{poly}_1(n,|T|,\frac{1}{\epsilon},\frac{1}{\delta})$ to $\text{poly}_1(n,\frac{1}{\epsilon},\frac{1}{\delta})$.

We consider a time series that starts from a point in which every variable is set to 0. For a suitable choice of n (any n such that $n-1$ is divisible by 3) we can build two 2-parent deterministic DBNs T_1 and T_2 over variables $X_1, ..., X_n$ with the following properties when started from a time slice with variables set to 0: in both T_1 and T_2, X_n remains 0 for $r \geq 2^{\frac{n-1}{3}}$ steps and then X_n goes to 1 at step $r+1$; in T_1 once X_n goes to 1 it remains 1; in T_2 when X_n goes to 1 it then reverts to 0 on the next step.

We choose $\epsilon = \delta = \frac{1}{4}$ and large enough n such that $2^{\frac{n-1}{3}} > \text{poly}_1(n,\frac{1}{\epsilon},\frac{1}{\delta})$. We present the algorithm L with a time series generated by T_1, of length r as specified in the previous paragraph, starting from the time slice in which all variables are set to 0. Then L must, with probability at least $1-\frac{1}{4}$, return a model that will correctly predict time slice $r+1$. Therefore, with probability at least $(\frac{3}{4})(\frac{3}{4}) > \frac{1}{2}$, L's output model predicts the value of X_n to be 1. Consider what happens when we give L exactly the same learning task, except that the target is T_2 instead of T_1. The time series of length r that L sees is identical to the previous one, so L will with probability greater than $\frac{1}{2}$ incorrectly predict the value of X_n at time slice $r+1$. Hence L will *not* produce, with probability at least $1-\delta$, a model that will predict time slice $r+1$ with accuracy at least $1-\epsilon$. □

3.3. *Stochastic Model of Boolean DBN from 2-slice data*

When we learn a Boolean DBN model, we are not only interested in learning the correct Boolean functions but also inferring a good model of probability wrt the target distribution. We therefore extend our theoretical framework to bring our model closer to practice. The foundation of this extension consists of the notions of *p-concept* (probabilistic concept) and (ϵ, γ)-*good models of probability*, defined as follows[8]. In these definitions X is the domain of possible examples and D is a probability distribution over X.

Definition 3.3. A *probabilistic concept (p-concept)* is a real-valued function $c : X \rightarrow [0, 1]$. When learning the p-concept c, the value $c(x)$ is interpreted as the probability that x exemplifies the concept being learned. A p-concept class $\mathcal{C}_p$ is a family of p-concepts. A learning algorithm for $\mathcal{C}_p$ attempts to learn a distinguished target p-concept $c \in \mathcal{C}_p$ wrt a fixed but unknown and arbitrary target distribution $\mathcal{D}$ over the instance space X[8].

Given this definition, it is easy to see that a function f_i in a (not necessarily deterministic) Boolean DBN, which gives the probability distribution over possible values for X_i at time $t + 1$ conditional on the values of the n variables at time t, is a p-concept. Therefore, a Boolean DBN as defined earlier is completely specified by a set of n p-concepts, one for each variable.

Definition 3.4. A p-concept h is an (ϵ, γ)-*good model of probability* of a target p-concept c with respect to D iff $\mathbf{Pr}_{x \in D}[|h(x) - c(x)| > \gamma] \leq \epsilon$.

We generalize this definition to apply to DBNs as follows. Given an input time slice, a DBN model defines a probability distribution over output time slices. We say that two DBNs M and T γ-disagree on an input time slice if they disagree by more than γ on the probability of an output time slice given that input. A learned DBN M is an (ϵ, γ)-*good model of probability* of a target DBN T with respect to a probability distribution D over input time slices if and only if the sum of the probabilities of input models on which M and T γ-disagree is at most ϵ.

The learning model we present next is a straightforward application of Kearns and Schapire's notion of *polynomially learnable with a model of probability* to DBNs, analogous to our earlier application of the PAC model to deterministic DBNs. In the following definition we take $\mathcal{C}_n$ to be any p-concept class. Thus for example $\mathcal{DBN}$(k-parents) is the set of DBNs in which each variable has at most k-parents; the p-concept class used here is the class of p-concepts of k-relevant variables, or the class of p-concepts representable by CPTs conditional on k parents.

Definition 3.5. Where $\mathcal{C}_n$ is a p-concept class, we say that an algorithm learns $\mathcal{DBN}(\mathcal{C}_n)$ with a model of probability iff there exist polynomials $\text{poly}_1(\_,\_,\_,\_,\_)$ and $\text{poly}_2(\_)$ such that for any target $T \in \mathcal{DBN}(\mathcal{C}_n)$, any $0 < \epsilon < 1$, $0 < \delta < 1$, and $0 < \gamma < 1$, and any probability distribution D over initial data points for time series: given $r \geq \text{poly}_2(n, |T|, \frac{1}{\epsilon}, \frac{1}{\delta}, \frac{1}{\gamma})$ data points, the algorithm runs in $\mathrm{poly}_2(rn)$ and with probability at least $1-\delta$ outputs an (ϵ,γ)-good model of probability of the target.

Theorem 3.3. *For any fixed $k \in \mathbb{N}$ the class $\mathcal{DBN}$(k-parents) is learnable with a model of probability from 2-slice time series data.*

Proof. We describe a learning algorithm B that is analogous to algorithm A of Theorem 3.1, and the correctness proof is analogous as well. In place of the kDNF learning algorithm used by the earlier algorithm A, algorithm B uses an algorithm P that with probability $1-\delta$ learns an (ϵ,γ)-good model of probability for any p-concept with at most k relevant variables[8].

More specifically, where δ, ϵ and γ are the parameters provided to algorithm B, algorithm B calls algorithm P using instead $\frac{\delta}{n}$, $\frac{\epsilon}{n}$ and $\frac{\gamma}{2n}$. Algorithm B iterates: for each variable $X_{i,2}$, $1 \le i \le n$, algorithm B makes a call to algorithm P with the examples and parameters as specified. Algorithm B's final model of probability for each X_i, $1 \le i \le n$, is $\Pr(X_{i,t}|X_{1,t-1},\ldots,X_{n,t-1}) = \Pr(X_{i,t}|\mathrm{Pa}(X_i)_{t-1})$, where $\mathrm{Pa}(X_i)_{t-1}$ denotes the (at most k) parents of X_i from the previous time step, as determined by algorithm P, and $\Pr(X_{i,t}|\mathrm{Pa}(X_i)_{t-1})$ denotes the specific function (representable as a CPT) learned by algorithm P.

Algorithm B runs in polynomial time since $n{*}\mathrm{poly}_1(n,|T|,\frac{n}{\epsilon},\frac{n}{\delta},\frac{2n}{\gamma})$ yields a polynomial, and each call to P runs in time polynomial in the size of its input. The remainder of the reasoning is analogous to that in the proof of Theorem 3.1, except that we must also note the following. If the learned DBN and target DBN agree within $\frac{\gamma}{2n}$ on the probability for a given setting for each variable X_i, $1 \le i \le n$, then they agree within γ on the probability of the entire setting. It follows that since *for any given variable X_i* the learned DBN with probability $1-\frac{\delta}{n}$ has an $(\frac{\epsilon}{n},\frac{\gamma}{2n})$-good model of probability compared with the target DBN, then with probability $1-\delta$ the learned DBN is an (ϵ,γ)-*good model of probability* of the target DBN. □

We can also extend our model from a *single time series* to apply to probabilistic concepts. Since deterministic DBNs are a special case of probabilistic ones, it follows that the result for learning $\mathcal{DBN}$(2-parents) with a model of probability, from a single, long time series is a negative one.

4. Lessons and Limitations

The results show that, for natural definitions of learnability, DBNs are learnable from 2-slice time series data but not from a single, long time series. If we adopt a compromise, with k-slice time series for fixed k greater than two, we can again get positive results but the total number of time slices, e.g., experiments to be run, increases linearly with k. Hence the results imply that while time series are desirable, they should be kept as short as possible. The reasoning is that if we subject a cell to many different conditions and collect multiple 2-slice data, we can capture the change easily, whereas if we subject a cell to 1 condition and collect one long time series one would have to wait a very long time to see this condition.

Because PAC bounds are worst-case, the number of examples they imply are required, while polynomial in the relevant parameters, can be much greater than typically required in reality. Nevertheless, we can gain some insight into which factors most affect sample size required for a given degree of accuracy. The sample sizes required by the algorithms in this paper follow directly from those required by the learning algorithms they employ as subroutines. The sample sizes for those algorithms grow linearly with the number of variables n, the target concept size, and $\frac{1}{\epsilon}$ (and $\frac{1}{\gamma}$ where relevant), and logarithmically with $\frac{1}{\delta}$. But note that the sizes of our target concepts in $\mathcal{DBN}$(kDNF) and $\mathcal{DBN}$(k-parents) are at least $O(n^k n)$, because we must specify the choice of k out of n possible parents for each of n variables. Therefore, by far the most important factor in sample size is k, and the next most important is n. Because current costs limit MS data set sizes to around 1000 experiments (we know of no such data sets), a value of three for k seems the largest reasonable value, with $k = 2$ probably more sensible. The size of the target concept can be limited based on prior knowledge about particular pathways in which we are most interested.

The models defined in this paper are a natural application of existing PAC models to DBN learning. Nevertheless, several assumptions are inherited in this application—some from PAC modeling and some from DBNs—and several additional assumptions have been made. We now discuss these classes of assumptions in turn.

Inherent to the use of PAC modeling are the assumptions that (1) we must perform well on *all* target concepts, and (2) examples are drawn randomly, independently according to some probability distribution. Regarding assumption (1), numerous regulatory motifs have been identified to date, including logic gates and memory elements[11,15,19], giving credence to the simple DBN representation of difficult target concepts such as counters. For some real biological pathways that have very short periods, perhaps single, long time series will be more effective than our results imply. Regarding assumption (2), it seems plausible that an organism's environment imposes some probability distribution over states in which its regulatory machinery may find itself, and to which it will need to respond. Nevertheless, perhaps through careful selection of experimental conditions, active learning approaches may arise that will benefit more from a few long time series than from many short ones.

Inherent in the use of DBNs are several notable assumptions as well. First, the DBN framework assumes we are modeling a *stationary process*; while the state of an organism is not static (e.g., mRNA levels may change over time), the organism's regulatory network itself does not change over time. This assumption appears reasonable for the application to learning regulatory pathways. But more specific assumptions include the assumption of discrete time steps—that an organism, like a computer, operates on a fixed clock that governs when changes occur—and a first-order Markov assumption, that the organism's next state is a function of only its previous state inputs. These assumptions clearly are violated to some extent, and those violations present caveats to our lessons. For example, perhaps collecting longer time series, with a very fast sampling rate, could allow our algorithms to try different sampling rates (by skipping some of the time steps), to find optimal rates for providing insight into certain processes.

Finally, we have made additional simplifying assumptions beyond those of the PAC framework or DBNs. Specifically, we have assumed all Boolean variables and no missing values or hidden variables. While discretization is common, one may also use the continuous values in the data. We see no obvious reason why using such values should change the lessons in this paper, but such a change is possible. Missing values are rare in MS data, but one might wish to include hidden variables for unmeasured environmental factors. Extending the present work to handle hidden variables is an interesting direction for further work.

In addition to handling continuous-valued variables and hidden variables, another significant direction for further work is the use of background knowledge to limit the space of potential models and hence the sample complexity. Also, based on the notion of *membership queries*, perhaps the models in this paper can be extended to model active learning approaches. Finally, we intend to use the lessons from this paper to design a large number of short time series experiments aimed at gaining more detailed models of a few key pathways in human cells.

References

1. T. Akutsu, S. Kuhara, O. Maruyama, and S. Miyano. Identification of gene regulatory networks by strategic gene disruptions and gene overexpressions. In *Proc. of 9th ACM-SIAM Symp. on Discrete Algs.*, pages 695–702, 1998.
2. T. Akutsu, S. Miyano, and S. Kuhara. Identification of genetic networks from

a small number of gene expression patterns under the boolean network model. *Proc. of PSB*, 4:17–28, 1999.

3. A. Bernard and A.J. Hartemink. Informative structure priors: Joint learning of dynamic regulatory networks from multiple types of data. In *PSB*, pages 459–470, 2005.
4. S. Dasgupta. The sample complexity of learning fixed-structure Bayesian networks. *MLJ*, 29:165–180, 1997.
5. N. Friedman and Z. Yakhini. On the sample complexity of learning bayesian networks. In *Proc. of UAI*, pages 274–282, 1996.
6. D. Husmeier. Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic bayesian networks. *Bioinformatics*, 19:2271–2282, 2003.
7. M. Kearns, M. Li, L. Pitt, and L.G. Valiant. On the learnability of boolean formulae. In *Proc. of ACM Theory of computing*, pages 285–295, 1987.
8. M. Kearns and R. Schapire. Efficient distribution-free learning of probabilistic concepts. In *Computational Learning Theory and Natural Learning Systems, Vol I: Constraints and Prospect*. MIT Press, 1994.
9. M.J. Kearns and U.V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, Cambridge, MA, 1994.
10. H. Lähdesmäki, I. Shmulevich, and O. Yli-Harja. On learning gene regulatory networks under the boolean network model. *MLJ*, 52:147–167, 2003.
11. H.H. McAdams and A.P. Arkin. Simulation of prokaryotic genetic circuits. *Ann. Rev. of Biophy. and Biomol. Structure*, 27:199–224, 1998.
12. I. Nachman, A. Regev, and N. Friedman. Inferring quantitative models of regulatory networks from expression data. *Bioinformatics*, 20:i248–i256, 2004.
13. I.M. Ong, J.D. Glasner, and D. Page. Modelling regulatory pathways in E.coli from time series expression profiles. *Bioinformatics*, 18:241S–248S, 2002.
14. D. Pe'er. Bayesian network analysis of signaling networks: A primer. *Science's STKE*, 281:pl4, 2005.
15. C.V. Rao and A.P. Arkin. Control motifs for intracellular regulatory networks. *Ann. Rev. of Biomed. Eng.*, 3:391–419, 2001.
16. K. Sachs, O. Perez, D. Pe'er, D.A. Lauffenburger, and G.P. Nolan. Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308:523–529, 2005.
17. I. Shmulevich, E.R. Dougherty, K. Seungchan, and W. Zhang. Probabilistic boolean networks: A rule-based uncertainty model for gene regulatory networks. *Bioinformatics*, 18:261–274, 2002.
18. L.G. Valiant. A theory of the learnable. *Comm. of ACM*, 27:1134–1142, 1984.
19. D.M. Wolf and A.P. Arkin. Motifs, modules and games in bacteria. *Curr. Op. in Microbio.*, 6:125–134, 2003.
20. Zhang Y., Wolf-Yadlin A., Ross P.L., Pappin D.J., Rush J., Lauffenburger D.A., and White F.M. Time-resolved mass spectrometry of tyrosine phosphorylatiopn sites in the EGF receptor signaling network reveals dynamic modules. *Mol. and Cell. Prot.*, 2005.