Computer Sciences Department

Learnability of Dynamic Bayesian Networks from Time Series Microarray Data

David Page
Irene Ong

Technical Report #1514

August 2004

UNIVERSITY OF WISCONSIN MADISON

Learnability of Dynamic Bayesian Networks from Time Series Microarray Data

David Page, Irene M. Ong
Department of Biostatistics & Medical Informatics and Department of Computer Sciences
University of Wisconsin Madison, WI 53706 USA
page@biostat.wisc.edu, ong@cs.wisc.edu

Abstract

Dynamic Bayesian networks (DBNs) are becoming widely used to learn gene regulatory networks from time series microarray data. Careful experimental design is required for data generation, because of the high cost of running each microarray experiment. This paper presents a theoretical analysis of learning DBNs without hidden variables from time series data. The analysis reveals, among other lessons, that under a reasonable set of assumptions a fixed budget is better spent on many short time series than on a few long time series.
Keywords: dynamic Bayesian networks, gene expression microarrays, gene regulatory networks, PAC-learnability, time series data

1 Introduction

Time series data, and dynamic Bayesian networks (DBNs) to model such data, are becoming widely used as an approach to learning gene regulatory networks (Ong et al., 2002; Husmeier, 2003; Perrin et al., 2003; Kim et al., 2003). In a typical experiment, gene expression microarrays are used to measure mRNA abundance *at several specific time points* after a particular stimulus to an organism or cell sample. The goal is then to learn a DBN that fits the time series data well. Biologists can visually examine this DBN structure and interpret an arc from gene X_1 at time t to gene X_2 at time $t+1$ as evidence that expression of gene X_1 influences expression of gene X_2 (Figure 1).

The most common alternative to DBNs for modeling gene expression data is to instead learn an ordinary Bayesian network (BN). A BN can be learned either from time series data (treating each time slice as a replicate) or from "ordinary" expression data, that is, data with one time point per stimulus or condition. However, there are several advantages that DBNs have over BNs. DBNs can construct cyclic regulations, whereas cycles are explicitly disallowed in BNs. DBN learning also can provide more insight into causality. For example, an induced arc from gene X_1 to gene X_2 in an ordinary BN simply means that the expression of gene X_1 is a good predictor of the expression of gene X_2 *at the same time* (Figure 2a). While this good prediction may be because expression of gene X_1 influences expression of gene X_2, it could just as easily be because expression of gene X_2 influences expression of gene X_1 or expression of both gene X_1 and gene X_2 are influenced by expression of another gene X_3 (Figure 2b). On the other hand, an induced arc from gene X_1 to gene X_2 in a DBN implies that expression of gene X_1 at one time slice is a consistently good predictor of gene X_2 at *the next time slice.* This good prediction

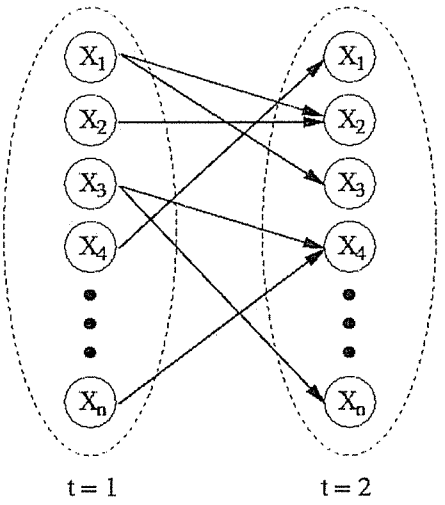


Figure 1: *Simple DBN model. Labeled circles within a dotted oval represent our variables in one time slice. Formally, arcs connecting variables from one time slice to variables in the next have the same meaning as in a BN, but they intuitively carry a stronger implication of causality. We note that in a DBN with more time slices, the arcs are always the same, e.g., the arc from* X_1 *at time slice 1 to* X_2 *at time slice 2 is also present from time slice* t *to time slice* $t+1$ *for all* $1 \leq t < T$ *where* T *is the last time slice in the model. This constancy of the arcs is justified by an assumption that the process being modeled is* stationary *though not static. While values of variables may change over time, the manner in which the value of one variable influences the value of a variable at the next time step (i.e., the parents and the conditional probability distribution for the latter variable) will not change.*

is unlikely to be because expression of gene X_2 influences expression of gene X_1; intuitively, it seems likely to be because expression of gene X_1 influences expression of gene X_2.[1]

Following earlier experiments in learning DBNs from time series microarray data (Ong et al., 2002), we have 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 microarrays, 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 microarrays r that will be run, and a given amount of time in which a DBN must be produced from this data, should we place a limit on the number of parents a node can have in the DBN and, if so, what should this limit be? One way to help answer these questions is to perform many runs with many time series data sets having different properties; unfortunately, at present 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 vari-

[1] An arc in a DBN does not establish causality definitively. Nevertheless, if a learned DBN contains arcs that imply novel potential causal relationships, in some cases biologists can test these novel relationships with additional, more focused (and time-consuming) experiments.

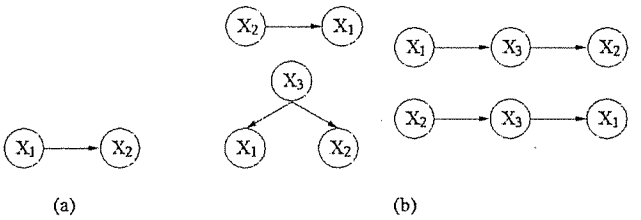


Figure 2: *(a) X_1 may be a good predictor of X_2, but is X_1* regulating *X_2? (b) Ground truth might be any one of these or a more complicated variant.*

ables, our DBNs also can be viewed as (deterministic or probabilistic) Boolean networks. In groundbreaking work Akutsu, Kuhara, Maruyama and Miyano (1998) formalized the task of constructing Boolean networks from gene expression microarray data. Further papers extended or improved their initial results (Akutsu et al., 1999; Akutsu et al., 2000; Shmulevich et al., 2002a; Shmulevich et al., 2002b; Lähdesmäki et al., 2003; Datta et al., 2003). Several of those papers provide formal results on the task of finding a consistent or best-fit Boolean network for the data. 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 follows in the spirit of this prior work on formal modeling of the Boolean network or DBN learning task, but it addresses the question of polynomial-time learnability using the PAC-learning framework (Valiant, 1984) and its extension to probabilistic concepts (Kearns & Schapire, 1994). 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, both positive and negative, imply the following practical advice for the design of time series microarray experiments and DBN learning algorithms.

First, many short time series experiments are preferable to a single long time series experiment. More specifically, time series experiments should be kept as short as possible, i.e. to two time steps. In other words, while time series experiments may be superior to individual independent experiments for determining causality, that message should not be carried to its logical extreme that the longer the time series the better.

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, two if possible. Even if we can fix part of the structure based on background knowledge of regulatory pathways, more than three parents to a node will likely yield very poor estimates of the relevant probabilities.

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 regulatory pathway, and we know most of the key players (genes). If in addition we have an overestimate of the potential other players, and there are l of these, then we can reduce this term further to l^2m, perhaps dramatically reducing the number of required examples, or microarrays.

The practical import of the preceding lessons of course depends on the fit between our formal models and the real world. The paper concludes by presenting caveats—some cases where these lessons may not apply—and by discussing possible extensions as directions for further work.

2 Definitions and Terminology

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

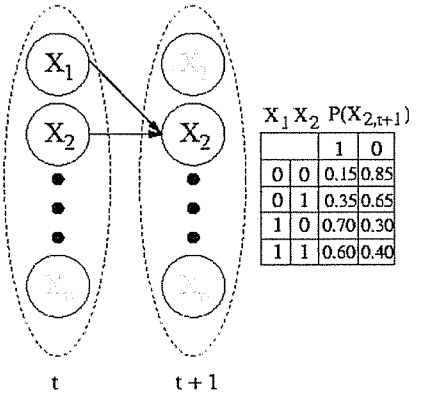
$$X_{1,1}, X_{2,1}, \ldots, X_{n,1},$$
$$X_{1,2}, X_{2,2}, \ldots, X_{n,2},$$
$$\ldots,$$
$$X_{1,T}, X_{2,T}, \ldots, X_{n,T}$$

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 $1 \leq i \leq n$. Note that the function 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 Figure 3. Or if the DBN is in fact 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 Figure 3 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) expression. The set of kDNF expressions is the set of all DNF expressions where each disjunct (conjunction) contains at most k literals.



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 3: *Example of probabilistic CPTs.*

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 have no data, e.g., variables that cannot be

observed or measured; such variables are also known as hidden variables. The present paper does not consider hidden variables or missing data, although those points are discussed later as a topic for further work.

In generating a gene expression microarray data set, the cost of r microarray experiments (measuring the expression of each gene in r samples of mRNA) is roughly 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 microarray 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. We later extend the definition to permit probabilistic 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 if and only if, given the input time slice, the two DBNs produce different outputs. A DBN model is $(1-\epsilon)$-accurate with respect to a target model if and only if 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. Given these preliminaries, the following definition is an application of the PAC-learning model to DBNs.

Definition 3.1. *An algorithm PAC-learns a deterministic $\mathcal{DBN}(\mathcal{C}_n)$ if and only if there exist polynomials $poly_1(\_,\_,\_,\_)$ and $poly_2(\_)$ 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 $r \geq poly_1(n, |T|, \frac{1}{\epsilon}, \frac{1}{\delta})$ data points, the algorithm runs in time $poly_2(rn)$ and with probability at least $1-\delta$ outputs a model that is $(1-\epsilon)$-accurate with respect to T.*

Theorem 3.2. *For any fixed $k \in \mathbb{N}$ the class of $\mathcal{DBN}(k\text{DNF})$ is PAC-learnable from 2-slice data.*

Proof. Our algorithm A learns one 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 (Kearns et al., 1987), we can run L on n feature vectors to find a concept in $\mathcal{C}_n$ that is consistent with our data.

Algorithm A iterates: for each variable $X_{i,2}$, $1 \leq i \leq n$, we make a call to 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 (that is, subsets of a probability space), then $\Pr(A \cup B) \leq \Pr(A)+\Pr(B)$ (Kearns & Vazirani, 1994). 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. By the definition of $\mathcal{DBN}(\mathcal{C}_n)$ this gives us $\Pr(Error_1 \cup Error_2 \cup \ldots \cup Error_n) = \Pr(Error_1)+\Pr(Error_2)+\ldots+\Pr(Error_n) = \frac{\epsilon}{n}+\frac{\epsilon}{n}+\ldots+\frac{\epsilon}{n} \leq \epsilon$, by the union bound. □

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 (because the space of allowed hypotheses is smaller), 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

In the previous subsection we showed that a Boolean DBN is PAC-learnable from 2-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 to fall between these two extremes. While the *total number of microarray experiments* r is determined largely by budget, the choice of time series *lengths* for any *fixed total* number of microarray experiments r usually is not driven by expense. Rather, researchers make the choice they believe will provide the most information, because r microarrays will have about the same cost regardless of whether they occur in one long time series or many shorter time series.

To gain some theoretical insight into whether a single, long time series might be more useful than many short 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 microarrays required might be less. Unfortunately, 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 an organism to an experimental condition and run a long time series of microarray expression 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. A positive result for single time series will be easier to obtain in this variant than in the original model.

Definition 3.3. *An algorithm learns a deterministic class* $\mathcal{DBN}(\mathcal{C}_n)$ *from a* single time series *if and only if 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 starting point for the time series: given*

a time series of any length $r \geq poly_1(n, |T|, \frac{1}{\epsilon}, \frac{1}{\delta})$*, the algorithm runs in time* $poly_2(rn)$ *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 is capable of performing well for most starting points, but only for the one given. For deterministic DBN models, which are all we are considering thus far, 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 $\mathcal{DBN}(k\text{-parents})$, the period is superpolynomial in n and the size of the target model, leading to the following negative result.

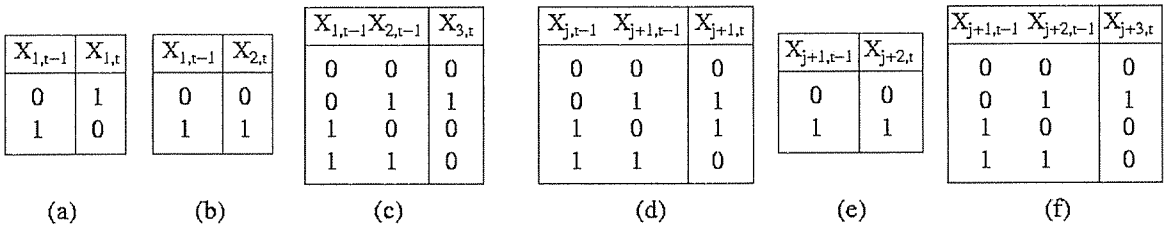
Theorem 3.4. *For any* $k \geq 2$ *the class of* $\mathcal{DBN}(k\text{-parents})$ *is not learnable from a single time series.*

Proof. Assume there exists a learning algorithm L for $\mathcal{DBN}(k\text{-parents})$. 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. Lemma 3.5 shows that for 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 $\frac{3}{4}$, 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 $1 - \epsilon$. □

Lemma 3.5. *There exists a 2-parent deterministic DBN over* j *variables with a period of* $2^{\frac{j}{3}}$*, for any positive integer* j *divisible by 3. Moreover, based on our specific construction we can build two 2-parent deterministic DBNs* T_1 *and* T_2 *over any variables* $X_1, ..., X_n$*,* $n = j + 1$ *for some* j *divisible by* 3*, with the following properties when started from a time slice with all 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.*



$X_{1,t-1}$	$X_{1,t}$
0	1
1	0

(a)

$X_{1,t-1}$	$X_{2,t}$
0	0
1	1

(b)

$X_{1,t-1}$	$X_{2,t-1}$	$X_{3,t}$
0	0	0
0	1	1
1	0	0
1	1	0

(c)

$X_{j,t-1}$	$X_{j+1,t-1}$	$X_{j+1,t}$
0	0	0
0	1	1
1	0	1
1	1	0

(d)

$X_{j+1,t-1}$	$X_{j+2,t}$
0	0
1	1

(e)

$X_{j+1,t-1}$	$X_{j+2,t-1}$	$X_{j+3,t}$
0	0	0
0	1	1
1	0	0
1	1	0

(f)

Figure 4: *Base case construction ((a),(b) and (c)) and inductive case construction ((d),(e) and (f)) of inductive proof of Lemma 3.5. Construction includes bit counter ((a),(d)), previous bit memory ((b),(e)) and 0-to-1 flag ((c),(f)).*

Proof. We begin with the first statement of the lemma. If we were allowed up to 3 parents per variable, we could represent a two-input (e.g., R-S or J-K) flip-flop (one parent for current state and two for inputs, to determine new state) with a single variable. We could then directly implement a standard up-counter from computer architecture with just two variables per bit. In our proof we still essentially implement an up-counter. But because we want to show a superpolynomial period can be obtained with just two parents per variable, we require three variables per bit of the up-counter.

Inductive hypothesis: Given j variables, a deterministic DBN exists with a period of $2^{\frac{j}{3}}$.

Base case: When $j = 3$, the proof is trivial as this can be done with 1 bit. Nevertheless, we employ the construction in Figure 4a to 4c so that every set of 3 variables is analogous to the others. The first variable is a 1-bit counter. The second variable, $X_{2,t}$, serves as a memory of the previous bit at the previous time slice, $X_{1,t-1}$. $X_{3,t}$ works as a flag, taking on the value 1 to indicate when X_1 has made the transition from 0 back to 1 on the previous time slice.

Inductive case: We prove that given $n = j + 3$ variables, a period doubling the period of $2^{\frac{j}{3}}$ can be generated. We first generate a counter that is based on the flag, $X_{j,t-1}$. If $X_{j,t-1}$ is 0, then $X_{j+1,t}$ takes on the value of $X_{j+1,t-1}$. Otherwise, when $X_{j,t-1}$ is 1, $X_{j+1,t}$ acts as a bit counter. $X_{j+2,t}$, serves as a memory of the previous bit at the previous time slice, $X_{j+1,t-1}$, mimicking its value in the previous time slice. Finally, $X_{j+3,t}$ works as a flag, taking on the value 1, to indicate when $X_{j+1,t-1}$ is 0 and $X_{j+2,t-1}$ is 1; and $X_{j+3,t}$ is 0 otherwise.

For the second statement of the lemma, $m = j + 1$, where j is divisible by 3. We construct a network as in the proof of the first statement of the lemma, such that X_j is 0 for the first $r \geq 2^{\frac{n-1}{3}}$ time steps, then becomes 1, and then immediately reverts to 0; that is, X_j is 1 just once every $2^{\frac{n}{3}}$ time steps. For T_1 we have a function for X_m such that $X_{m,t}$ is 1 if and only if $X_{j,t-1}$ (that is, $X_{m-1,t-1}$) is 1 *or* $X_{m,t-1}$ is 1. For T_2 we have a function for X_m such that $X_{m,t}$ is 1 if and only $X_{j,t-1}$ (that is, $X_{m-1,t-1}$) is 1. □

3.3 Stochastic Model of Boolean DBN from 2-slice data

In our first model of learning Boolean DBN models from 2-slice data, we made a simplifying assumption that the DBN models were deterministic, in keeping with the standard assumption for target concepts in the PAC model. In reality, DBNs are probabilistic models, with deterministic DBNs as simply a special case. 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 with respect to the target distribution. We therefore now extend our theoretical framework and to bring our model closer to practice. The foundation of this extension consists of the no-

tions of *p-concept* (probabilistic concept) and (ϵ, γ)-*good models of probability*, defined as follows (Kearns & Schapire, 1994). In these definitions X is the domain of possible examples and D is a probability distribution over X.

Definition 3.6. *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* $c \in \mathcal{C}_p$ *with respect to a fixed but unknown and arbitrary target distribution* $\mathcal{D}$ *over the instance space X (Kearns & Schapire, 1994).*

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 in this paper is completely specified by a set of n p-concepts, one for each variable.

Definition 3.7. *A p-concept h is an* (ϵ, γ)-good model of probability *of a target p-concept c with respect to D if and only if* $\mathbf{Pr}_{x \in D}[|h(x) - c(x)| > \gamma] \leq \epsilon$.

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 3.8. *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 if and only if there exist polynomials* $poly_1(\_, \_, \_, \_, \_)$ *and* $poly_2(\_)$ *such that for any target concept* $T \in \mathcal{DBN}(\mathcal{C}_n)$, *any* $0 < \epsilon < 1$, $0 < \delta < 1$, *and* $0 < \gamma < 1$, *and any probability distribution D over initial data points for time series: given* $r \geq poly_2(n, |T|, \frac{1}{\epsilon}, \frac{1}{\delta}, \frac{1}{\gamma})$ *data points, the algorithm runs in time* $poly_2(rn)$ *and with probability at least* $1 - \delta$ *outputs an* (ϵ, γ)-*good model of probability of the target.*

Theorem 3.9. *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 the algorithm A of Theorem 3.2, and the correctness proof is analogous as well. 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 $X_{1,1}, \ldots, X_{n,1}$. 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 (Kearns & Schapire, 1994). The learned p-concept for variable X_i can be expressed as a set of at most k parents for variable X_i and a CPT for variable X_i given those parents.

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 $X_{i,2}$, $1 \leq i \leq n$, 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, $1 \leq i \leq n$, 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 3.2, 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, $1 \leq i \leq n$, 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 extend also our model of learning from a *single time series* to apply to probabilistic concepts in a similar fashion. But 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 proven in this paper 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., microarrays to be run, increases linearly with k. Hence the results in this paper imply that while time series are desirable, they should be kept as short as possible. We return to discuss limits on this lesson after discussing lessons related to sample size.

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 are required in reality. Nevertheless, we can gain some insight from these numbers 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 microarray data set sizes to around one thousand microarrays (in fact, we currently

know of no data sets quite that large), 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 further contained if we limit our models to trying to predict only a small subset of variables in terms of another small subset, based on background knowledge about particular regulatory pathways in which we are most interested.

Of course this entire discussion hinges upon an acceptance that the learning models defined in this paper are reasonable. The models themselves 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), perhaps particularly difficult target concepts such as counters, while representable in simple DBNs, do not appear in biological regulatory pathways. If in fact all real biological pathways 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. Inappropriate sampling rates have been noted as a potential source of error when modeling time series microarray data (Bay et al., 2003). Our results do not speak to uses such as this for longer time series.

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 with microarray data, one may discretize to more than two values or may use the continuous values originally reported in a microarray experiment. 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 microarray data, but one might wish to include hidden variables for unmeasured environmental factors or for other players in a regulatory pathway, such as certain proteins that may act as transcription factors or for other types of signaling. 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 microarray experiments aimed at gaining more detailed models of a few key regulatory pathways in human cells.

References

Akutsu, T., Kuhara, S., Maruyama, O., & Miyano, S. (1998). Identification of gene regulatory networks by strategic gene disruptions and gene overexpressions. *Proc. the 9th Annual ACM-SIAM Symposium on Discrete Algorithms* (pp. 695–702).

Akutsu, T., Miyano, S., & Kuhara, S. (1999). Identification of genetic networks from a small number of gene expression patterns under the boolean network model. *Pacific Symposium on Biocomputing, 4*, 17–28.

Akutsu, T., Miyano, S., & Kuhara, S. (2000). Inferring qualitative relations in genetic networks and metabolic pathways. *Bioinformatics, 16*, 727–734.

Bay, S., Chrisman, L., Pohorille, A., & Shrager, J. (2003). Temporal aggregation bias and inference of causal regulatory networks. *IJCAI-03 Workshop on Learning Graphical Models for Computational Genomics* (pp. 58–65).

Datta, A., Choudhary, A., Bittner, M. L., & Dougherty, E. R. (2003). External control in markovian genetic regulatory networks. *Machine Learning, 52*, 169–191.

Husmeier, D. (2003). Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic bayesian networks. *Bioinformatics, 19*, 2271–2282.

Kearns, & Schapire (1994). Efficient distribution-free learning of probabilistic concepts. In S. J. Hanson, G. A. Drastal and R. L. Rivest (Eds.), *Computational learning theory and natural learning systems, volume i: Constraints and prospect*, vol. 1. Bradford, MA: MIT Press.

Kearns, M., Li, M., Pitt, L., & Valiant, L. G. (1987). On the learnability of boolean formulae. *Proceedings of the nineteenth annual ACM conference on Theory of computing* (pp. 285–295). New York, NY: ACM Press.

Kearns, M. J., & Vazirani, U. V. (1994). *An introduction to computational learning theory.* Cambridge, MA: MIT Press.

Kim, S., Imoto, S., & Miyano, S. (2003). Inferring gene networks from time series microarray data using dynamic bayesian networks. *Briefings in Bioinformatics, 4*, 228–235.

Lähdesmäki, H., Shmulevich, I., & Yli-Harja, O. (2003). On learning gene regulatory networks under the boolean network model. *Machine Learning, 52*, 147–167.

Ong, I. M., Glasner, J. D., & Page, D. (2002). Modelling regulatory pathways in E.coli from time series expression profiles. *Bioinformatics, 18*, 241S–248S.

Perrin, B. E., Ralaivola, L., Mazurie, A., Bottani, S., Mallet, J., & D'Alche-Buc, F. (2003). Gene networks inference using dynamic bayesian networks. *Bioinformatics, 19*, II138–II148.

Shmulevich, I., Dougherty, E. R., Seungchan, K., & Zhang, W. (2002a). Probabilistic boolean networks: A rule-based uncertainty model for gene regulatory networks. *Bioinformatics, 18*, 261–274.

Shmulevich, I., Saarinen, A., Yli-Harja, O., & Astola, J. (2002b). Inference of genetic regulatory networks under the best-fit extension paradigm. In W. Zhang and I. Shmulevich (Eds.), *Computational and statistical approaches to genomics.* Boston: Kluwer Academic Publishers.

Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM, 27*, 1134–1142.