Probably Approximately Correct Learning of Regulatory Networks from Time-Series Data

Arthur Carcano[1], François Fages[2(✉)], and Sylvain Soliman[2]

[1] Ecole Normale Supérieure, Paris, France
arthur.carcano@ens.fr
[2] Inria, University Paris-Saclay, Lifeware Group, Palaiseau, France
{Francois.Fages,Sylvain.Soliman}@inria.fr

Abstract. Automating the process of model building from experimental data is a very desirable goal to palliate the lack of modellers for many applications. However, despite the spectacular progress of machine learning techniques in data analytics, classification, clustering and prediction making, learning dynamical models from data time-series is still challenging. In this paper we investigate the use of the Probably Approximately Correct (PAC) learning framework of Leslie Valiant as a method for the automated discovery of influence models of biochemical processes from Boolean and stochastic traces. We show that Thomas' Boolean influence systems can be naturally represented by k-CNF formulae, and learned from time-series data with a number of Boolean activation samples per species quasi-linear in the precision of the learned model, and that positive Boolean influence systems can be represented by monotone DNF formulae and learned actively with both activation samples and oracle calls. We consider Boolean traces and Boolean abstractions of stochastic simulation traces, and study the space-time tradeoff there is between the diversity of initial states and the length of the time horizon, and its impact on the error bounds provided by the PAC learning algorithms. We evaluate the performance of this approach on a model of T-lymphocyte differentiation, with and without prior knowledge, and discuss its merits as well as its limitations with respect to realistic experiments.

1 Introduction

Modelling biological systems is still an art which is currently limited in its applications by the number of available modellers. Automating the process of model building is thus a very desirable goal to attack new applications, develop patient-tailored therapeutics, and also design experiments that can now be largely automated with a gain in both the quantification and the reliability of the observations, at both the single cell and population levels.

Machine learning is revolutionising the statistical methods in biological data analytics, data classification and clustering, and prediction making. However, learning dynamical models from data time-series is still challenging. A recent survey on probabilistic programming [14] highlighted the difficulties associated

J. Feret and H. Koeppl (Eds.): CMSB 2017, LNBI 10545, pp. 74–90, 2017.
DOI: 10.1007/978-3-319-67471-1_5

with modelling time, and concluded that existing frameworks are not sufficient in their treatment of dynamical systems. There has been early work on the use of machine learning techniques, such as inductive logic programming [19] combined with active learning in the vision of the "robot scientist" [4], to infer gene functions, metabolic pathway descriptions [1,2] or gene influence systems [3], or to revise a reaction model with respect to CTL properties [5]. Since a few years, progress in this field is measured on public benchmarks of the "Dream Challenge" competition [15,18]. Logic Programming, and especially *Answer Set Programming* (ASP), provide efficient tools such as CLASP [11] to implement learning algorithms for Boolean models. They have been applied in [12] to the detection of inconsistencies in large biological networks, and have been subsequentially applied to the inference of gene networks from gene expression data and to the design of discriminant experiments [26]. Furthermore, ASP has been combined with CTL model-checking in [20] to learn mammalian signalling networks from time series data, and identify erroneous time-points in the data.

Active learning extends machine learning with the possibility to call oracles, e.g. make experiments, and budgeted learning adds costs to the calls to the oracle. The original motivation for the budgeted learning protocol came from medical applications in which the outcome of a treatment, drug trial, or control group is known, and the results of running medical tests are each available for a price [8]. In this context, multi-armed bandit methods [7] currently provide the best strategies. In [16], a bandit-based active learning algorithm is proposed for experiment design in dynamical system identification.

In this paper, we consider the framework of Probably Approximately Correct (PAC) Learning which was introduced by Leslie Valiant in his seminal paper on a theory of the learnable [24]. Valiant questioned what can be learned from a computational viewpoint, and introduced the concept of PAC learning, together with a general-purpose polynomial-time learning protocol. Beyond the algorithms that one can derive with this methodology, Valiant's theory of the learnable has profound implications on the nature of biological and cognitive processes, of collective and individual behaviors, and on the study of their evolution [25].

Here we present PAC learning as a possible basis to develop a method for the automated discovery of influence models of biochemical processes from time-series data. To the best of our knowledge, the application of PAC learning to dynamical models of biochemical systems has not been reported before. We show that Thomas' gene regulatory networks [22,23] can be naturally represented by Boolean formulae in conjunctive normal forms with a bounded number of litterals (i.e. k-CNF formulae), and can be learned from Boolean traces with a number of Boolean transition samples per species quasi-linear in the precision of the learned model, using Valiant's PAC learning algorithm for k-CNF formulae. We also show that Boolean influence systems with their positive Boolean semantics discussed in [9] can be naturally represented by monotone DNF formulae, and learned actively from a set of positive samples with calls to an oracle.

For the sake of evaluation, we consider Boolean traces and Boolean abstractions of stochastic simulation traces, and study the space-time tradeoff there is

between the diversity of initial states and the length of the time horizon, and its impact on the error bounds provided by PAC learning algorithms. In the following, we first illustrate our results[1] with a toy example, the Lotka-Volterra prey-predator system as running example, and then on a Thomas regulatory network of the differentiation of the T-helper lymphocytes from [17,21], composed of 32 influences and 12 variables. We evaluate the performance of PAC learning on this model, with and without prior knowledge, and discuss its merits as well as its limitations with respect to realistic experiments.

2 Preliminaries on PAC Learning

2.1 PAC Learning Protocol

Let n be the dimension of the model to learn, and let us consider a finite set of Boolean variables $x_1, \ldots, x_n$, A vector is an assignment of the n variables to $\mathbb{B} = \{0, 1\}$; A Boolean function $G : \mathbb{B}^n \to \mathbb{B}$; assigns a Boolean value to each vector. The idea behind the PAC learning protocol is to discover a Boolean function[2], G, which approximates a hidden function F, while restricting oneself to the two following operations:

- SAMPLE(): returns a positive example, i.e. a vector v such that $F(v) = 1$. The output of SAMPLE() is assumed to follow a given probability distribution $D(v)$, which is used to measure the approximation of the result.
- ORACLE(v): returns the value of $F(v)$ for any input vector v.

Definition 1 ([24]). *A class $\mathcal{M}$ of* Boolean functions *is said to be* learnable *if there exists an algorithm $\mathcal{A}$ with some precision parameter $h \in \mathbb{N}$ such that:*

- *$\mathcal{A}$ runs in polynomial time both in n and h;*
- *for any function F in $\mathcal{M}$, and any distribution D on the positive examples, $\mathcal{A}$ deduces with probability higher than $1 - h^{-1}$ an approximation G of F such that*
  - *$G(v) = 1$ implies $F(v) = 1$ (no false positive)*
  - $\sum_{v \text{ s.t. } F(v)=1 \wedge G(v)=0} D(v) < h^{-1}$ *(low probability of false negatives)*

Note that it is possible to use two different parameters h_1 and h_2 for the *probability of false negatives* and the *quality of the approximation*, but here, we used $h_1 = h_2 = h$ for the sake of simplicity.

[1] For the sake of reproducibility, the code used in this article is available at http://lifeware.inria.fr/wiki/software/#CMSB17.

[2] More generally, the PAC learning protocol can discover partial vectors, but for the applications discussed in the current article it is enough to only consider total vectors.

2.2 PAC Learning Algorithms

Valiant showed the learnability of some important classes of functions in this framework, in particular for Boolean formulae in conjunctive normal forms with at most k literals per conjunct (k-CNF), and for monotone (i.e. negation free, positive literals only) Boolean formulae in disjunctive normal form (DNF).

The computational complexity of the PAC learning algorithms for these classes of functions is expressed in terms of a function $L(h, S)$, defined as the smallest integer i such that in i independent Bernoulli trials, each with probability at least h^{-1} of success, the probability of having fewer than S successes is less than h^{-1}. Interestingly, this function is *quasi-linear in h and S*, more precisely for all integers $S \geq 1$ and reals $h > 1$, we have $L(h, S) \leq 2h(S + \log_e h)$ [24].

Theorem 1 ([24]). *For any k, the class of k-CNF formulae on n variables is learnable with an algorithm that uses $L(h, (2n)^{k+1})$ positive examples and no call to the oracle.*

The proof is constructive and relies on Algorithm 1 below. In this algorithm, the initialization of the learned function g to the false constraint expressed as the conjunction of all possible *clauses* (i.e. disjunctions of litterals) leads to the learning of a minimal generalization of the positive examples with no false positive and low probability of false negatives.

Algorithm 1. PAC-learning of k-CNF formulae.

1. initialise g to the conjunction of all the $(2n)^k$ possible clauses of at most k literals,
2. do $L(h, (2n)^{k+1})$ times
   (a) $v :=$ SAMPLE()
   (b) delete all the clauses in g that do not contain a literal true in v
3. output: g

In our implementation of the PAC-learning algorithm for k-CNF formulae, we shall make use of the lattice structure of k-clauses ordered by implication. Interestingly, this data structure allows for

- $O(1)$ access to any k-clause;
- and for a clause c, $O(1)$ access to the smallest clauses implied by c and to the biggest clauses that imply c.

The class of monotone DNF formulae is also learnable. Let the *degree* of a Boolean formula be the largest number of prime implicants (i.e., minimal formulae covering one of the product-terms of the Boolean formula expressed as a sum of products) in an equivalent rewriting of the formula as a non-redundant sum of prime-implicants.

Theorem 2 ([24]). *The class of monotone DNF formulae on n variables is also learnable with an algorithm that uses $L(h, d)$ examples and dn calls to the oracle, where d is the* degree *of the function to learn.*

The proof relies on Algorithm 2. As previously, the algorithm guarantees that a minimal generalization is learned from both the samples and the oracle. The polynomial computational complexity follows from the fact that each monomial m is a prime implicant of f by construction, and that it is constructed by at most n calls to the oracle.

Algorithm 2. PAC-learning of monotone DNF formulae.

1. initialise g with false (constant zero),
2. do $L(h, d)$ times
   (a) $v :=$ SAMPLE()
   (b) if $v \Rightarrow g$ exit
   (c) for $i :=$ 1 *to* n
      i. if x_i is determined in v
         A. $v^* := v[x_i \leftarrow *]$
         B. if ORACLE(v^*) then
            – $v := v^*$
            – $m := \bigwedge_{v \Rightarrow x_j} x_j \wedge \bigwedge_{v \Rightarrow \neg x_k} \neg x_k$
            – $g := g \vee m$
3. output: g

3 Influence Models of Molecular Cell Processes

In this section, we present the formalism of influence systems used to model regulatory networks in cell molecular biology. We assume again a finite set of molecular species $\{x_1, \ldots, x_n\}$ and consider Boolean states that represent the activation or presence of each molecular species of the system, i.e. vectors in $\mathbb{B}^n$ that specify whether or not the ith species is present, or the ith gene activated.

3.1 Influence Systems with Forces

Influence systems with forces have been introduced in [9] to generalize the widely used logical models of regulatory networks *à la* Thomas [22], in order to provide them with a hierarchy of semantics including quantitative differential and stochastic semantics, similarly to reaction systems [10].

Definition 2 ([9]). *An* influence system I *is a set of quintuples* (P, I, t, σ, f) *called* influences*, noted in the examples below in Biocham v4*[3] *syntax,* `f for P/I -> t` *if* $\sigma = +$*, and* `f for P/I -< t` *if* $\sigma = -$*, where*

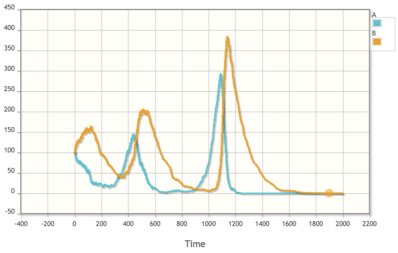
[3] http://lifeware.inria.fr/biocham4.

- P *is a multiset on* S*, called* positive sources *of the influence,*
- I *a multiset of* negative sources*,*
- $t \in S$ *is the* target*,*
- $\sigma \in \{+, -\}$ *is the* sign *of the influence, accordingly called either* positive or negative influence*,*
- *and* $f : {\mathbb{R}_+}^n \to \mathbb{R}_+$ *is a function*[4] *called the* force *of the influence.*

The positive sources are distinguished from the negative sources of an influence (positive or negative), in order to annotate the fact that in the differential semantics, the source increases or decreases the force of the influence, and in the Boolean semantics with negation whether the source, or the negation of the source, is a condition for a change in the target.

Example 1. The classical birth-death model of Lotka–Volterra can be represented by the following influence system between a proliferating prey A and a predator B:

```
k1 * A * B for A, B -< A.
k1 * A * B for A, B -> B.
k2 * A for A -> A.
k3 * B for B -< B.
```



The influence forces can be used for differential or stochastic simulation as above. This example contains both positive and negative influences but no influence inhibitor, i.e. no negative source in the influences: $(\{A, B\}, \emptyset, A, -, k1 * A * B)$, $(\{A, B\}, \emptyset, B, +, k1 * A * B)$, $(\{A\}, \emptyset, A, +, k2 * A)$ and $(\{B\}, \emptyset, B, -, k3 * B)$. For an example of influence with inhibitor, one can consider the specific inhibition of the proliferation rate of A by some variable C (which is distinguished from a general negative influence of C on A) by writing C as an inhibitor of the positive influence of A on A: `k2 * A/(1 + C) for A/C -> A.`

Definition 3 (Boolean Semantics). *The Boolean semantics (resp. positive Boolean semantics) of an influence system* $\{(P_i, I_i, t_i, \sigma_i, f_i)\}_{1 \leq i \leq n}$ *over a set* S *of* n *variables, is the Boolean transition system* $\longrightarrow$ *defined over Boolean state vectors in* $\mathbb{B}^n$ *by* $\boldsymbol{x} \longrightarrow \boldsymbol{x}'$ *if there exists an influence* $(P_i, I_i, t_i, \sigma_i, f_i)$ *such that* $\boldsymbol{x} \models \bigwedge_{p \in P_i} p \bigwedge_{n \in I_i} \neg n$ *(resp.* $\boldsymbol{x} \models \bigwedge_{p \in P_i} p$*) and* $\boldsymbol{x}' = \boldsymbol{x}\ \sigma_i\ t_i$*.*

where adding (resp. subtracting) t amounts to making the corresponding coordinate true (resp. false).

Equivalently, the Boolean semantics of an influence system over n species, $x_1, \ldots, x_n$, can be represented by n activation and n deactivation Boolean functions, which determine the possible transitions from each Boolean state:

[4] More precisely, in a well-formed influence system, f is assumed to be partially differentiable; $x_i \in P$ if and only if $\sigma = +$ (resp. $-$) and $\partial f / \partial x_i(\boldsymbol{x}) > 0$ (resp. < 0) for some value $\boldsymbol{x} \in \mathbb{R}_+^n$; and $x_i \in I$ if and only if $\sigma = +$ (resp. $-$) and $\partial f / \partial x_i(\boldsymbol{x}) < 0$ (resp. > 0) for some value $\boldsymbol{x} \in \mathbb{R}_+^n$.

Definition 4 (Boolean Activation Functions). *The* Boolean activation functions ${x_k}^+, {x_k}^- : \{0,1\}^n \to \{0,1\}$, $1 \leq k \leq n$, *of an influence system* $\mathcal{M}$ *are*

$${x_k}^+ = \bigvee_{(P,I,x_k,+,f)\in\mathcal{M}} \bigwedge_{p\in P_i} p \bigwedge_{n\in I_i} \neg n \qquad {x_k}^- = \bigvee_{(P,I,x_k,-,f)\in\mathcal{M}} \bigwedge_{p\in P_i} p \bigwedge_{n\in I_i} \neg n$$

The positive activation functions *are defined without negation by ignoring the inhibitors.*

Conversely any system of Boolean activation functions can be represented by an influence system by putting the activation functions in DNF, and associating an influence to each conjunct.

Note that the positive Boolean semantics simply ignores the negative sources of an influence. This is motivated by the abstraction and approximation relationships that link the Boolean semantics to the stochastic semantics and to the differential semantics, for which the presence of an inhibitor decreases the force of an influence but does not prevent it to apply [9].

Definition 5 (Stochastic Semantics). *The stochastic semantics (resp. positive stochastic semantics) of an influence system* $\{(P_i, I_i, t_i, \sigma_i, f_i)\}_{1\leq i\leq n}$ *over a set* S *of* n *variables, relies on the transition system* $\longrightarrow$ *defined over discrete states, i.e. vectors in* $\mathbb{N}^n$, *by* $\forall (P_i, I_i, t_i, \sigma_i, f_i), \boldsymbol{x} \longrightarrow \boldsymbol{x'}$ *with propensity* f_i *if* $\boldsymbol{x} \geq P_i, \boldsymbol{x} < I_i$ *(resp. no condition on* I_i*) and* $\boldsymbol{x'} = \boldsymbol{x}\ \sigma_i\ t_i$. *Transition probabilities between discrete states are obtained through normalization of the propensities of all enabled transitions, and the time of next transition is given by an exponential distribution* [13].

We call a positive influence system, an influence system without inhibitors or interpreted under the positive semantics.

3.2 Monotone DNF Representation of Positive Influence Systems

Definition 4 shows how to represent an influence system by $2 * n$ activation functions in DNF, and *positive* influence systems by *monotone* DNF activation functions.

Example 2. The activation functions of the Lotka–Volterra influence system of Example 1 are monotonic DNF formulae with only one conjunct since in this example there is only one signed influence per variable:

$$A^+ = (A) \qquad B^+ = (A \wedge B)$$
$$A^- = (A \wedge B) \qquad B^- = (B)$$

3.3 k-CNF Representation of General Influence Systems

Monotone DNF formulae cannot encode the Boolean dynamics of influence systems with negation, which tests the absence of inhibitors, i.e., negative literals. This is possible using a k-CNF representation of the activation functions, provided that there are at most k species that can play a given "role". For instance, in a hypothetic activation function in CNF $(a \vee b \vee c) \bigwedge (d \vee e) \bigwedge \neg f$, each clause can be interpreted as a role, and each role can be played by a limited number of species, at most k.

Example 3. The activation functions of the prey-predator model with inhibition of Example 1 cannot be represented by monotone formulae. They can however be represented by the following 1-CNF formulae ($k = 1$ since there is only one positive and one negative influence for each target):

$$A^+ = (A) \wedge (\neg C) \qquad A^- = (A) \wedge (B)$$
$$B^+ = (A) \wedge (B) \qquad B^- = (B)$$

Example 4. In Sect. 5, we shall study a model of T lymphocyte differentiation which contains 2-CNF activation functions, for instance

$$\text{IFNg}^+ = (\text{STAT4} \vee \text{TBet}) \qquad \text{IFNg}^- = (\neg\text{STAT4}) \wedge (\neg\text{TBet})$$

3.4 k-CNF Models of Thomas Functional Influence Systems

Definition 6 ([22]). *A* Thomas *network on a finite set of genes* $\{x_1, \ldots, x_n\}$ *is defined by* n *Boolean functions* $\{f_1, \ldots, f_n\}$ *which give for each gene its possible next state, given the current state.*

The difference with the previous general influence systems is that the activation and deactivation functions are exclusive and defined by one single function. As shown in [9], non-terminal self-loops cannot be represented in Thomas functional influence systems. Given a general influence system with activation functions ${x_i}^+$ and ${x_i}^-$, one can associate a Thomas network with attractor function[5]

$$f_i(v) = \begin{cases} 1 \text{ if } \begin{cases} v_i = 0 \text{ and } {x_i}^+(v) = 1 \\ v_i = 1 \text{ and } {x_i}^-(v) = 0 \end{cases} \\ 0 \text{ if } \begin{cases} v_i = 0 \text{ and } {x_i}^+(v) = 0 \\ v_i = 1 \text{ and } {x_i}^-(v) = 1 \end{cases} \end{cases}$$

k-CNF formulae can again be used to represent Thomas gene regulatory network functions with some reasonable restrictions on their connectivity. In particular, it is worth noticing that in Thomas networks of degree bounded by k, each gene has at most k regulators, each gene activation function f_i thus depends of at most k variables and can consequently be represented by a k-CNF formula.

[5] Note that this function ignores the cases where $v_i = 0$ and ${x_i}^-(v) = 0$, or $v_i = 1$ and ${x_i}^+(v) = 1$ which may create loops in non-terminal states in general influence systems.

Example 5. The above translation applied to Example 1 gives $f_A = A \wedge \neg B, f_B = 0$. Note that the form of f_B means that the only possible state change for B is from 1 to 0.

Example 6. The T-lymphocyte model studied in Sect. 5 is originally a Thomas' network, where we have, for instance: $f_{\text{IFNg}} = (\text{STAT4} \vee \text{TBet})$

4 PAC Learning from Traces

4.1 Diverse Initial States Versus Long Time Horizon

In practice, one cannot assume to have full access to the hidden Boolean function as required by Sample and Oracle, but rather to data time-series, or traces, produced from biological experiments. For the scope of this paper, we consider simulation traces obtained from a hidden model which we wish to discover. Two types of traces are considered: Boolean and stochastic simulation traces. In both cases, the mapping to the concepts of PAC-learning is easy: a Sample for the activation function x^+ (resp. deactivaction function x^-) is a state s_i such that $x_i < x_{i+1}$ (resp. $x_i > x_{i+1}$). See Fig. 1 for an example.

$$\cdots \rightarrow \underset{(a)}{\begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}} \rightarrow \underset{(b)}{\begin{pmatrix} \boldsymbol{1} \\ 1 \\ 0 \\ 1 \end{pmatrix}} \rightarrow \underset{(c)}{\begin{pmatrix} 1 \\ 1 \\ 0 \\ \boldsymbol{0} \end{pmatrix}} \rightarrow \cdots$$

Fig. 1. Illustration of a Boolean trace with three steps. Between a and b, the first gene has been activated, and between b and c, the last one has been deactivated.

One striking feature of PAC learning is to associate a guarantee on the quality h of each learnt Boolean function depending on the number of samples used, namely $L(h, (2n)^{k+1})$, where n is the number of genes/molecules observed, and k is the maximum number of literals per conjunct. In practice, k seems to be limited to 3 or 2, and the number $2n$ of different possible literals in a clause, can also be reduced through prior knowledge (e.g. in Sect. 5.3).

It is worth noticing that the global guarantee on the learnt model is the minimum of all precision bounds h. In order to perfectly recover a hidden model, it is thus necessary to have sufficiently diverse samples. For this reason, one should expect to get better performance with large sets of short traces obtained from a uniformly distributed set of initial states, rather than with a small set of long traces which introduce a bias in the distribution of the transition samples (e.g. when looping in an attrator). The important point is that PAC learning algorithms do provide bounds on the error according to this space-time trade-off.

On the other hand, the ORACLE procedure needs to evaluate the (de)activation function on a given vector v, that is, it needs to be able to set the system in a state abstracted by v and say whether or not a given gene can be (de)activated from this state. In practice, this cannot be achieved without approximation. The intuitive solution would be to set the system in the desired state and see whether or not the gene is (de)activated. However, different atomic steps are possible from a given state and we have no guarantee that the one we are interested in will happen in a given finite number of runs. These considerations militate for studying an extension of the PAC-learning framework with an oracle that would be only probabilistic.

4.2 PAC Learning from Boolean Traces

A first experiment was to produce Boolean (de)activation traces by simulation of a given influence model, and use them to learn the hidden model. Figure 2 reports our results obtained with 25 Boolean traces of short length equal to 2 (i.e. when trading time for space) on Example 1, where to increase readability we used long names for the species. It is worth noticing that in this particular model, the positive infuences cannot be learned from (de)activation traces, since they contain their target as positive source and thus do not correspond to an activation function. Indeed, the activation functions in the Lokta–Volterra models report the apparition on extinction of the species' population as a whole and not of individuals of it. The results in this tradeoff are perfect in the sense that the negative influences are correctly inferred.

```
Prey, Predator -< Prey.
Prey, Predator -> Predator.
Prey -> Prey.
Predator -< Predator.
```

```
Predator -< Predator
Predator, Prey -< Prey
```

Fig. 2. The Lokta-Volterra prey vs. predator influence model of Example 1 with long names (left panel) and the (most likely) influence model PAC-learned on 25 simulations of length 2 (right panel) from random initial states.

```
Predator+ : False
Predator- : Predator /\ !Prey
Prey+ : False
Prey- : Predator /\ Prey
```

```
Predator / Prey -< Predator
Predator, Prey -< Prey
```

Fig. 3. Most likely PAC-learned activation functions (left pane, where `!A` stands for $\neg A$), and corresponding influence model (rigth panel obtained by CNF-DNF conversion), on a *single random Boolean trace of length 50* from the standard initial state with prey and predator present.

On the other hand, PAC learning from a single Boolean trace obtained from the standard initial state where both the prey and the predator are present (i.e. trading space for time), most likely leads to the influence model shown in Fig. 3. For the prey to go extinct, there must be both a prey in the first place and a predator to eat it. This is correct. For the predator to disappear, it is necessary that there is a predator in the first place and that there is no prey. The first part of this conjunction is true, but the second is false: predators may disappear even if there are preys left. However, this case is unlikely, the most likely case is that the predator will go extinct only once there are no more preys left for it to eat. As can be seen even on this very simple example, the "approximately" in PAC has a precise meaning. Yet, as explained in Definition 1, the quantification of this approximation relies on the knowledge of the distributions of the samples. In the present case, the probability of a positive example v of (de)activation function $x\pm$ to be sampled is strongly and intuitively correlated to both the probability that the system reaches state v and the probability of the actual (de)activation of gene x from state v.

4.3 PAC Learning from Stochastic Traces

Let us now consider sets of stochastic traces. They can be produced from an influence system with forces, using Gillespie's algorithm (Definition 5), assuming here mass-action kinetics with rate 1 for all influences. The initial states are random, but with equal probability to be 0 or > 0 in order to facilitate the observation of the inhibitions in the influences. The states in $\mathbb{N}^n$ can be

```
biocham: pac_learning('library:examples/lotka_volterra/LVi.bc
    ', 50, 1).
% Maxmimum K used:  minimum number of samples for h=1: 18

% 14 samples (max h ~ 0.7777777777777778)
Predator -< Predator

% 7 samples (max h ~ 0.3888888888888889)
Predator,Prey -> Predator

% 1 samples (max h ~ 0.05555555555555555)
Predator,Prey -< Prey

% 21 samples (max h ~ 1.1666666666666667)
Prey -> Prey
```

Listing 1: Biocham running the k-CNF PAC learning algorithm on the Lotka–Volterra influence model from stochastic simulation traces of length 1, obtained from 50 random initial states. Among those 50 initial states, 7 had both prey and predator absent, leading to no sample.

abstracted to Boolean samples by the usual $\{0, > 0\}$ abstraction for the states, and the increasing/decreasing abstraction for choosing samples for the activation/deactivation functions. Using the same $\{0, > 0\}$ abstraction to detect samples would again forbid to learn autocatalytic influences like `Prey -> Prey` for the same reason as in the Boolean case.

Interestingly, Listing 1 shows that here again, even with a low number of samples, and therefore a very low precision bound h, one can find the full model with less than 50 simulations of length 1, all starting from random initial states.

5 Evaluation on a Model of T-Helper Lymphocytes Differentiation

5.1 Boolean Thomas Network

In this section we evaluate the performance of the k-CNF PAC learning algorithm on an influence system of 12 variables and 32 influences that models the differentiation of the T-helper lymphocytes. This model, presented in [21] is actually a Boolean simplification of the original multi-level model of [17]. It studies the regulatory network of stimuli leading to differentiation between Th-1 and Th-2 lymphocytes from an original CD4+ T helper (Th-0). The model has three different stable states corresponding to Th-0 (naive lymphocyte), Th-1 and Th-2 when IL12 is off, and two others when IL12 is on (the Th-0 one is lost). Figure 4 shows the influence graph of the model. The influence model is given in Listing 2.

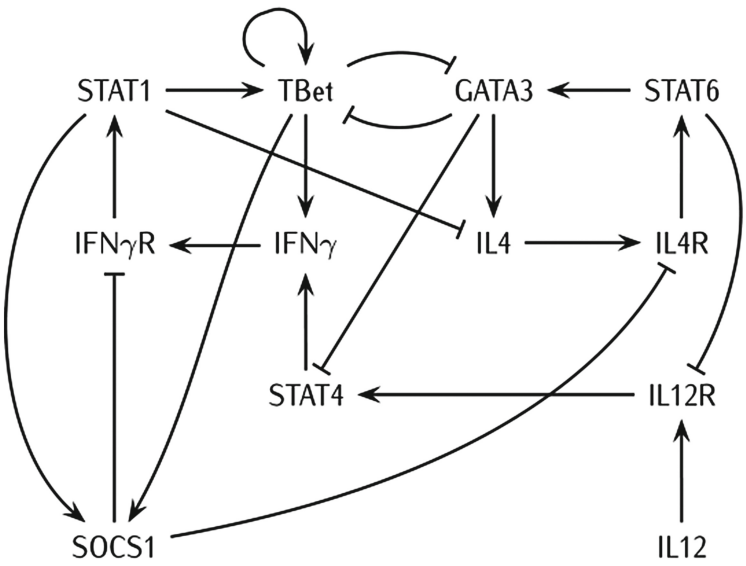


Fig. 4. Figure 4 of [21] displaying the Th-lymphocyte differentiation model.

```
STAT4, TBet -> IFNg.
/ STAT4 -< IFNg.
/ TBet -< IFNg.
GATA3 / STAT1 -> IL4.
/ GATA3 -< IL4.
STAT1 -< IL4.
IFNg / SOCS1 -> IFNgR.
/ IFNg -< IFNgR.
SOCS1 -< IFNgR.
IL4 / SOCS1 -> IL4R.
/ IL4 -< IL4R.
SOCS1 -< IL4R.
IL12 / STAT6 -> IL12R.
/ IL12 -< IL12R.
STAT6 -< IL12R.
IFNgR -> STAT1.
/ IFNgR -< STAT1.
IL4R -> STAT6.
/ IL4R -< STAT6.
IL12R / GATA3 -> STAT4.
/ IL12R -< STAT4.
GATA3 -< STAT4.
STAT1 -> SOCS1.
TBet -> SOCS1.
/ STAT1, TBet -< SOCS1.
STAT6 / TBet -> GATA3.
STAT1 / GATA3 -> TBet.
TBet / GATA3 -> TBet.
GATA3 -< TBet.
/ STAT1, TBet -< TBet.
/ STAT6 -< GATA3.
TBet -< GATA3.
```

Listing 2: Influence system for the lymphocyte differentiation of Example 5.

All learning experiments described below run on a 3 GHz Linux desktop in less than 3 s. However, the CNF (activation functions) to DNF (influence model) conversions could be very slow, reaching more than 4 min in the worst cases (e.g. with a single simulation of 10^6 steps). Note also that since IL12 is an input, in all experiments the PAC learning algorithm only finds $false$ as Boolean function for its activation or deactivation. We thus removed it from the results below for readability.

5.2 *Ab initio* PAC Learning from Stochastic Traces

When using stochastic simulations in this example, Fig. 5 shows that a simple randomization of the initial states (while keeping the total number of samples constant) provides a much more homogeneous repartition of activation and deactivation samples (as shown by the decreasing standard deviation), and, as expected, a much higher confidence h in the learnt model. The minimum number of samples gives in fact a quasi-linear estimate of the model confidence h. Obviously, more diverse initial states reveal more about the model structure than longer experiments.

On the other hand, the error measured as the number of false positive and false negative influences (right scale divided by 10), reveals a non monotonic behavior: in this example, there is a zone with sets of 10 to 100 traces, where PAC learning produces models with very complex (de)activation formulae that are not readable by humans and that produce many errors. Above 500 traces from random initial states the learnt model is perfect.

The guarantee on the accuracy of the learnt model comes directly from Valiant's work with the approximation bounds. Note however that Valiant's results stand only if we actually have at least L samples for each of the

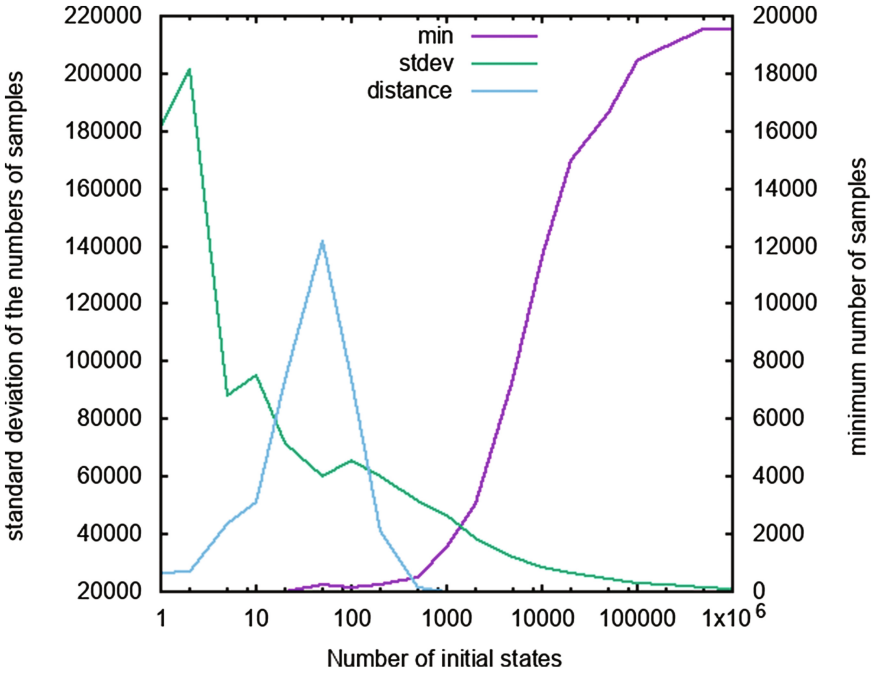


Fig. 5. Minimum numbers of (de)activation samples with standard deviations, and model errors (i.e. false positive and negative influences) obtained for the 24 boolean functions of the Th-lymphocyte example, as a function of the number of initial states (with total number of samples kept constant by adjusting the time horizon).

(de)activation functions, where L is Valiant's bound. A first naive approach might be to simply let the trace run for $2nL$ steps, or a constant factor of it. Nevertheless, the repartition of samples for each function can be pretty non-uniform. Interestingly, the minimal number of samples gives us the lowest $L(h, S)$ and thus quasi-linearly the lowest guarantee h.

Our simulation results show that when using PAC-learning to find the structure of a regulatory model, an approach based on mutants (knock-offs, over-expression, etc.) is much more informative than an approach based on (repeated or longer) similar observations. Note that this is in line with the pratice of integrative analyses such as [6] and its more than 130 mutants.

5.3 PAC Learning with Prior Knowledge on the Influence Graph

Furhtermore, to improve the guarantee h and the corresponding accuracy of the learnt models, especially for bigger models, it is necessary to look again at what constrains h. We have $samples = 2h(S + \log h)$, where S is the number of possible k-CNF clauses, bounded by $(2n)^{k+1}$.

The previous section explored the diversity of the samples, another option is to reduce S for a given n. This can be done by formalizing possible/known

interactions as prior knowledge, as is common in Machine Learning, effectively restricting the possible clauses for each activation/deactivation function.

Here we want the user to be able to specify, for each gene x, a set of gene V_x which are the only ones on which x^+ and x^- may depend. If one views the influences as a graph, this is akin to specifying a set of possible (undirected) edges outside of which the algorithm cannot build its influence system. An example of such hints for the lymphocyte model are given at http://lifeware.inria.fr/wiki/software/#CMSB17.

In such an example, the number of possible clauses becomes bounded by 3^3 (maximum 3 effectors that are either a positive literal, a negative literal or not in the clause at all) instead of 26^4 (Valiant's bound). Since for a given number of samples h varies quasi-linearly in S, the improvement is drastic (50000 times less samples for the same h). The accuracy of the model is, as expected with such guarantee, improving a lot.

6 Conclusion and Perspectives

We have shown that Valiant's work on PAC learning provides an elegant trail, with error bounds, to attack the challenge of inferring the structure of influence models from the observation of data time series, and more precisely to automatically discover possible regulatory networks of a biochemical process, given sufficiently precise observations of its executions.

The Boolean dynamics of biochemical influence systems, including Thomas regulatory networks, can be represented by k-CNF formulae without loss of generality, and k-CNF PAC learning algorithm can be used to infer the structure of the network, and bound the errors made according to the distribution of the state transition samples and the space-time tradeoff in the traces. When dimension increases, we have shown on an example of T-lymphocyte differentiation from the literature that the k-CNF PAC learning algorithm can also leverage available prior knowledge on the system to deliver precise results with a reasonable amount of data.

The Boolean dynamics of positive influence systems can also be straightforwardly represented by monotone DNF activation and deactivation functions, and monotone DNF PAC learning algorithm applied with an interesting recourse to oracles which are particularly relevant in the perspective of online active learning and experimental design. More work is needed however to make comparisons on common benchmarks with other approaches already investigated in this context, such as Answer Set Programming (ASP) and budgeted learning, and to investigate the applicability of these methods to real experiments taking into account the noise in the observations.

Acknowledgements. This work is partly supported by the ANR project Hyclock.

References

1. Angelopoulos, N., Muggleton, S.H.: Machine learning metabolic pathway descriptions using a probabilistic relational representation. Electron. Trans. Artif. Intell. **7**(9), 1–11 (2002). also in Proceedings of Machine Intelligence
2. Angelopoulos, N., Muggleton, S.H.: Slps for probabilistic pathways: Modeling and parameter estimation. Technical Report TR 2002/12. Department of Computing, Imperial College, London, UK (2002)
3. Bernot, G., Comet, J.P., Richard, A., Guespin, J.: A fruitful application of formal methods to biological regulatory networks: Extending Thomas' asynchronous logical approach with temporal logic. J. Theor. Biol. **229**(3), 339–347 (2004)
4. Bryant, C.H., Muggleton, S.H., Oliver, S.G., Kell, D.B., Reiser, P.G.K., King, R.D.: Combining inductive logic programming, active learning and robotics to discover the function of genes. Electron. Trans. Artif. Intell. **6**(12), 1–36 (2001)
5. Calzone, L., Chabrier-Rivier, N., Fages, F., Soliman, S.: Machine learning biochemical networks from temporal logic properties. In: Priami, C., Plotkin, G. (eds.) Transactions on Computational Systems Biology VI. LNCS, vol. 4220, pp. 68–94. Springer, Heidelberg (2006). doi:10.1007/11880646_4
6. Chen, K.C., Calzone, L., Csikász-Nagy, A., Cross, F.R., Györffy, B., Val, J., Novàk, B., Tyson, J.J.: Integrative analysis of cell cycle control in budding yeast. Mol. Biol. Cell **15**(8), 3841–3862 (2004)
7. Deng, K., Bourke, C., Scott, S.D., Sunderman, J., Zheng, Y.: Bandit-based algorithms for budgeted learning. In: ICDM (2007)
8. Deng, K., Zheng, Y., Bourke, C., Scott, S., Masciale, J.: New algorithms for budgeted learning. Mach. Learn. **90**, 59–90 (2013)
9. Fages, F., Martinez, T., Rosenblueth, D.A., Soliman, S.: Influence systems vs Reaction systems. In: Bartocci, E., Lio, P., Paoletti, N. (eds.) CMSB 2016. LNCS, vol. 9859, pp. 98–115. Springer, Cham (2016). doi:10.1007/978-3-319-45177-0_7
10. Fages, F., Soliman, S.: Abstract interpretation and types for systems biology. Theor. Comput. Sci. **403**(1), 52–70 (2008)
11. Gebser, M., Kaufmann, B., Neumann, A., Schaub, T.: *clasp*: A conflict-driven answer set solver. In: Baral, C., Brewka, G., Schlipf, J. (eds.) LPNMR 2007. LNCS (LNAI), vol. 4483, pp. 260–265. Springer, Heidelberg (2007). doi:10.1007/978-3-540-72200-7_23
12. Gebser, M., Schaub, T., Thiele, S., Usadel, B., Veber, P.: Detecting inconsistencies in large biological networks with answer set programming. In: Garcia de la Banda, M., Pontelli, E. (eds.) ICLP 2008. LNCS, vol. 5366, pp. 130–144. Springer, Heidelberg (2008). doi:10.1007/978-3-540-89982-2_19
13. Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. J. Phys. Chemis. **81**(25), 2340–2361 (1977)
14. Gordon, A.D., Henzinger, T.A., Nori, A.V., Rajamani, S.K.: Probabilistic programming. In: Proceedings of the on Future of Software Engineering, FOSE 2014, pp. 167–181, NY, USA. ACM, New York (2014)
15. Hill, S.M., et al.: Inferring causal molecular networks: empirical assessment through a community-based effort. Nat. Method. **1**(4), 310–318 (2016)
16. Llamosi, A., Mezine, A., dÁlché-Buc, F., Letort, V., Sebag, M.: Experimental design in dynamical system identification: a bandit-based active learning approach. In: Calders, T., Esposito, F., Hüllermeier, E., Meo, R. (eds.) ECML PKDD 2014. LNCS, vol. 8725, pp. 306–321. Springer, Heidelberg (2014). doi:10.1007/978-3-662-44851-9_20

17. Mendoza, L.: A network model for the control of the differentiation process in Th cells. Biosystems **84**(2), 101–114 (2006)
18. Meyer, P., Cokelaer, T., Chandran, D., Kim, K.H., Loh, P.R., Tucker, G., Lipson, M., Berger, B., Kreutz, C., Raue, A., Steiert, B., Timmer, J., Bilal, E., Sauro, H.M., Stolovitzky, G., Saez-Rodriguez, J.: Network topology and parameter estimation: from experimental design methods to gene regulatory network kinetics using a community based approach. BMC Syst. Biol. **8**(1), 1–18 (2014)
19. Muggleton, S.H.: Inverse entailment and progol. New Gener. Comput. **13**, 245–286 (1995)
20. Ostrowski, M., Paulevé, L., Schaub, T., Siegel, A., Guziolowski, C.: Boolean network identification from perturbation time series data combining dynamics abstraction and logic programming. Biosystems **149**, 139–153 (2016)
21. Remy, E., Ruet, P., Mendoza, L., Thieffry, D., Chaouiya, C.: From logical regulatory graphs to standard petri nets: dynamical roles and functionality of feedback circuits. In: Priami, C., Ingólfsdóttir, A., Mishra, B., Riis Nielson, H. (eds.) Transactions on Computational Systems Biology VII. LNCS, vol. 4230, pp. 56–72. Springer, Heidelberg (2006). doi:10.1007/11905455_3
22. Thomas, R.: Boolean formalisation of genetic control circuits. J. Theor. Biol. **42**, 565–583 (1973)
23. Thomas, R.: Regulatory networks seen as asynchronous automata : a logical description. J. Theor. Biol. **153**, 1–23 (1991)
24. Valiant, L.: A theory of the learnable. Commun. ACM **27**(11), 1134–1142 (1984)
25. Valiant, L.: Probably Approximately Correct. Basic Books (2013)
26. Videla, S., Konokotina, I., Alexopoulos, L.G., Saez-Rodriguez, J., Schaub, T., Siegel, A., Guziolowski, C.: Designing experiments to discriminate families of logic models. Front. Bioeng. Biotechnol. **3**, 131 (2015)