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

Arthur Carcano, François Fages, Sylvain Soliman

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, France
Francois.Fages@inria.fr, 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 quasi linear number of Boolean activation samples per species, and that positive Boolean influence systems can be represented by monotone DNF formulae and learned actively with both activation samples and oracle calls. 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 [13] highlighted the difficulties associated 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 [17] 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 [16]. Logic Programming, and especially *Answer Set Programming* (ASP), provide efficient tools such as CLASP [10] to implement learning algorithms for Boolean models. They have been applied in [11] 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 [24]. Furthermore, ASP has been combined with CTL model-checking in [18] 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 [7]. In this context, multi-armed bandit methods [6] currently provide the best strategies. In [14], 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 [22]. 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 [23].

Here we investigate PAC learning as 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 [21,20] 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 transitions with a quasi linear number of Boolean transition samples, using Valiant's PAC learning algorithm for k-CNF formulae. We also show that Boolean influence systems with their positive Boolean semantics discussed in [8] can be naturally represented by monotone DNF formulae, and learned actively from a set of positive samples with calls to an oracle.

In the following, these results[3] are first illustrated with a toy example, the Lotka-Volterra prey-predator system as running example, and then evaluated

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

on a Boolean influence model of the differentiation of the T-helper lymphocytes from [19,15], composed of 32 influences and 12 variables.

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 total vector is a Boolean assignment, in $\mathbb{B} = \{0, 1\}$; A Boolean function $G : \mathbb{B}^n \to \mathbb{B}$; assigns a Boolean value to each total vector; A concept $F : {\mathbb{B}_*}^n \to \mathbb{B}$ assigns a Boolean value to each vector.

The idea behind the PAC learning protocol is to discover a concept, or a Boolean function, G which approximates a hidden concept 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 ([22]). *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)*

For the sake of simplicity, the same precision parameter h is used above for quantifying both the *probability* that the result is correct, and the *approximation* error tolerated in the correctness criterion. For the scope of this paper, we could have also removed the distinction between concepts and Boolean functions since in the sequel we will learn Boolean functions only, and obtain samples from data time series which provide total vectors. The distinction was kept here for the sake of generality.

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) 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 the 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, i.e. for all integers $S \geq 1$ and reals $h > 1$, we have $L(h, S) \leq 2h(S + \log_e h)$ [22].

Theorem 1 ([22]). *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 Alg. 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 mainly 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 in an equivalent rewriting of the formula as a non-redundant sum of prime-implicants.

Theorem 2 ([22]). *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 Alg. 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. total 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 [8] to generalize the widely used logical models of regulatory networks *à la* Thomas [20], in order to provide them with a hierarchy of semantics including quantitative differential and stochastic semantics, similarly to reaction systems [9].

Definition 2 ([8]). *An* influence system I *is a set of quintuples* (P, I, t, σ, f) *called* influences*, noted in the examples below in Biocham v4*[4] *syntax,* `f for P/I -> t` *if* $\sigma = +$*, and* `f for P/I -< t` *if* $\sigma = -$*, where*

- 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*[5] *called the* force *of the influence.*

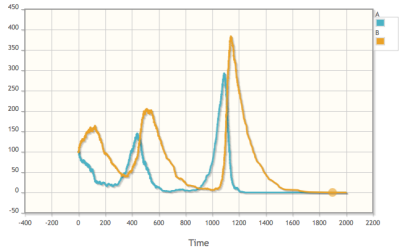
[4] `http://lifeware.inria.fr/biocham4`

[5] 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 $\boldsymbol{x} \in \mathbb{R}_+^n$.

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 total 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 $\boldsymbol{x}'=\boldsymbol{x}\ \sigma_i\ t_i$.*

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:

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*

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

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

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

Conversely any 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 [8].

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 [12].*

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

Def. 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 Ex. 1 are monotonic DNF formulae with only one conjunct since in this example there is only one signed influence per variable:

$$\begin{aligned} A^+ &= (A) \\ A^- &= (A \wedge B) \\ B^+ &= (A \wedge B) \\ B^- &= (B) \end{aligned}$$

3.3 k-CNF Representation of General Influence Systems

Monotone DNF formulae cannot encode the Boolean dynamics of influence systems with negation, which test the absence of inhibitors. 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-perdator model with inhibition of Ex. 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):

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

Example 4. In Sec. 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 ([20]). *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 [8], 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[6]

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

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.

Example 5. The above translation applied to Ex. 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.

[6] 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 6. The T-lymphocyte model sutided in Sec. 5 is originally a Thomas' network, where we have, for instance:

$$f_{\mathrm{IFNg}} = (\mathrm{STAT4} \vee \mathrm{TBet})$$

4 PAC Learning from Traces

4.1 Steps and Traces

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 either from real biological experiments, or from simulations, like here for the purpose of evaluating the learning method. More specifically, and as illustrated in Fig. 1, we consider that our experimental setup enables us to identify a "step" of its evolution, i.e. that we are able to obtain a trace $(v_i)_{0 \leq t \leq T}$ of the Boolean state of activation of its species where for all t in $[0, T-1]$, v_t and v_{t+1} differ in exactly one coordinate. That is, the state of exactly one species has changed. Even though we will finally not require it, we first also consider that our experimental setup enables us to put the system in a given state, i.e. that for any $v \in \mathbb{B}^n$ we can set the experiment's initial conditions so that they are correctly abstracted by v.

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

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

4.2 Sample and Oracle on Traces

Sampling. It is worth remarking that sampling of activation and deactivation functions is straightforward in this setting. Referring to Fig. 1, (a) is a positive example for the activation function ${x_1}^+$, and (c) for ${x_4}^-$. A call to Sample can then simply be a search in the trace for the next positive example for the current function. Issues may arise if there are not enough samples to guarantee a good approximation. We assume here that those issues can be solved by running several traces from different initial states. This very point is discussed in more details on an example in Section 5.2

Oracle. The ORACLE function needs to evaluate the (de)activation function on a given total 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. The intuitive solution is the following: 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 several runs of the experiment. Moreover, in its simple form presented above, Valiant's framework does not allow for the oracle to be probabilistic. In practice however, since we cannot run this "state and observe the first atomic step" scheme an infinite number of time to be able to tell that the (de)activation can only happen with probability 0, we will have to content ourself with an imperfect oracle.

4.3 PAC Learning from Boolean Traces

A first experimentation was to simulate Boolean traces for a given influence network, and use them as a basis to learn. We report results on our running Ex. 1., but to increase readability we used long names for the species (populations) as shown in Listing 1. The results are presented in Listing 2. (in the output `!A` stands for $\neg A$).

```
{"Predator": 1, "Prey": 1}
Predator, Prey -> Predator (100)
Predator, Prey -< Prey  (100)
Prey -> Prey (1)
Predator -< Predator (1)
```

Listing 1: The Lokta-Voltera prey vs. predator model of Ex. 1 with explicit names, forces (numbers in parenthesis) and initial state (first line).

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

Listing 2: Results of PAC-learning on traces of the Boolean simulation of the Lokta–Voltera example.

The inferred model can be interpreted as follows. Both the predator and the prey species cannot appear. It is important to note that the activation functions in the Lokta–Voltera models reports the apparition on extinction of the species'

population as a whole and not of individuals of it. For the predator to disappear, it is necessary that there is a predator in the first place and that there is no prey. If the first part of this conjunction is obviously true, the second is false: strictly speaking, predators may disappear even if there are preys left, yet this case is very unlikely: the most likely case is that the predator will go extinct only once there are no more preys left for it to eat. This is a good example of how the learning is only approximate. Finally, 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.

As can be seen even on this very simple example, the "approximately" in PAC has a precise meaning. Yet, as explained in Def. 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.

4.4 PAC Learning from Stochastic Traces

Let us now consider stochastic traces produced from the hidden model, using Gillespie's algorithm for the simulation of the stochastic semantics given in Def. 5.

The traces obtained are in $\mathbb{N}^n$, they are then abstracted to Boolean traces over $\mathbb{B}^n$ by the usual $\{0, > 0\}$ abstraction. It would also be possible to try and learn directly a Boolean model from stochastic traces (using $x_i \longrightarrow x_i + 1$ (resp. $x_i - 1$) as positive example for the activation (resp. deactivation)), but since the abstraction of the state remains necessary we present here a fully abstracted method.

We ran two example reactions, one is given in Listing 3, and the other using again the Lotka–Volterra model of Listing 1 but with 10 `Predators` and 1000 `Preys` as initial state. As indicated in Listing 4, the first example found the original model.

The Lotka–Volterra model leads to the same approximation as in the Boolean case, for the same reason, the very low probability of extinction of the predator when there are still preys alive. Note that this could be circumvented with a specific initial state (a single predator) and repeated simulations. A more detailed discussion of this topic is given in Section 5.2.

As previously, the prey-predator example (Listing 5) highlights how dependent to the traces the learned model is. To further detail this, it is necessary to dive into the Markov chain of the prey-predator stochastic process. A simplification of it is given in Fig. 2. One can remark that state (2) is an equilibrium state, whereas state (0) is terminal. Because most of the experiments will start in state (3), we can highlight two possible paths, A and B, leading respectively to (2) and (0). Because (de)activation (and hence sample) only comes from a changing of node in Fig. 2, each of these pathways result in a different resulting model.

```
{}
-> A
-> E
A / E -> B
B -> C
A -< A
B -< B
C -< C
E -< E
```

Listing 3: Test model, A and E appear naturally in the medium, and A can be turned into B in absence of E. B can be turned into C. All of the species can disappear due to dilution.

```
A+ : !A
A- : A
E+ : !E
E- : E
B+ : A /\ !E /\ !B
B- : B
C+ : B /\ !C
C- : C
```

Listing 4: Results for the test example

For A this is:

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

For B this is:

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

Both those learned model are approximate, as the sampling will by design miss some of the possible positive examples because only one of A and B paths can be followed. Indeed, the only way to get the correct result is to run several time the experiment, and hope that probabilities favor us enough so that we have samples from pathways A and B.

Result of such run will then be correct, as below:

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

Listing 5: Results for the Prey-Predator model

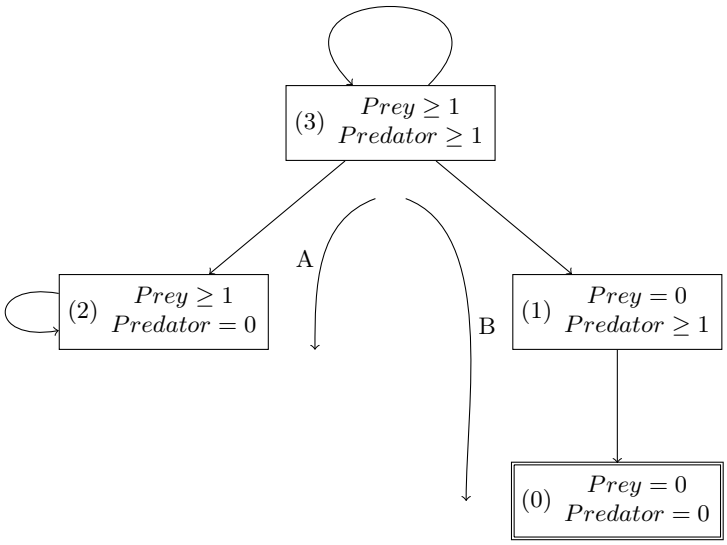


Fig. 2. A simplification of the Markov chain of the Lokta-Voltera process. Probabilities are not given but all arrows have non-null probability to be taken.

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

5 Evaluation on a Model of T-helper Lymphocytes Differentiation

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. All learning experiments described below were run on a low-end laptop in less than 3s.

5.1 Boolean Thomas Network

This model, presented in [19] is actually a Boolean simplification of the original multi-level model of [15]. 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).

Fig. 3 shows the influence graph of the model. The corresponding code is given in appendix (Listing 6).

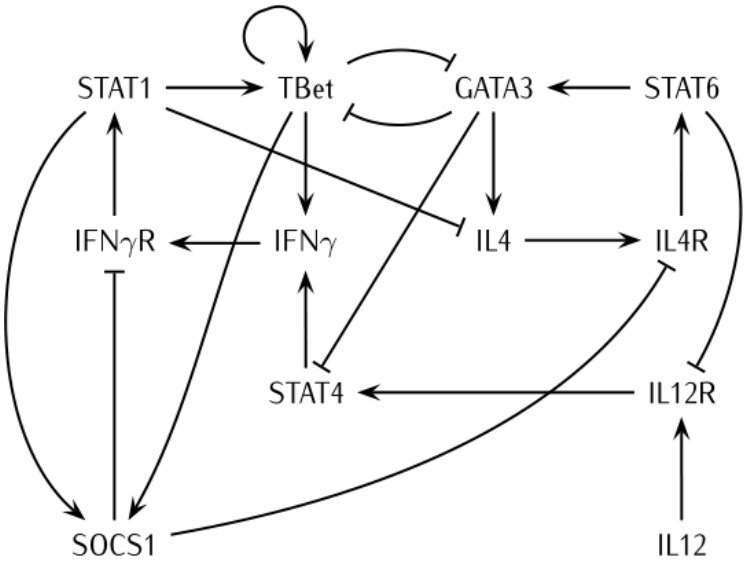


Fig. 3. Fig. 4 of [19] displaying the Th-lymphocyte differentiation model.

5.2 *Ab initio* PAC Learning from Boolean Traces

Results of learning on the traces resulting from the Boolean simulation of this system from one single initial state gave very poor results. By this, we mean not that they were bad at predicting the behavior of the model (because guarantees on this come directly from Valiant's work) but that they were based on not very informative traces. Valiant's results stand only if we actually have at least L samples for each of the (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. Unfortunately, the repartition of samples for each function can be pretty non-uniform, as illustrated in Fig. 4. Hence, we would have to find a system to track the number of samples collected so far for each function, but without running infinitely long shall one gene be never (de)activated.

This entices us to find way to make the learning more precise. A first possibility is to aggregate samples resulting from multiple runs, done from random starting points. For example 140 runs provides a significant improvement, as the output is now more or less readable, as displayed in Listing 7. Note that randomizing does not really uniformize the repartition of the samples, as shown in Fig. 5. We can also remark that – for the same number of steps – there is not necessarily more *different* samples when randomizing periodically the state (ie, restarting with random conditions) than when not doing it, as shown Fig. 6. Yet, the quality of learning is better when doing periodic random reinitialization, leading us to believe that even if there is not more different samples, they are distributed in a way that cover more exhaustively the behavior of the system..

5.3 PAC Learning with Prior Knowledge on the Influence Graph

Even with such improved results, the obtained model remains approximate, and it becomes obvious that with much bigger examples, the amount of data needed to get back the initial model would become huge. Machine learning with prior knowledge can save a lot of work by constraining the possible results and pruning the search.

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. Example of such hints for the lymphocyte model are given in Listing 8.

It is remarkable than when given such prior knowledge, the model becomes fully readable for the experimenter, and provides them with information on the exact structure of the influences, as shown in Listing 9.

6 Conclusion and Perspectives

We have shown that Valiant's work on PAC learning provides an elegant trail 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 strucutre of the network from a sufficiently large and diverse set of state transition traces. When dimension increases, we have shown on an example of T-lymphocyte differentiation from the litterature 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 particular biological technologies.

References

1. Angelopoulos, N., Muggleton, S.H.: Machine learning metabolic pathway descriptions using a probabilistic relational representation. Electronic Transactions in Artificial Intelligence 7(9) (2002), also in Proceedings of Machine Intelligence 19
2. Angelopoulos, N., Muggleton, S.H.: Slps for probabilistic pathways: Modeling and parameter estimation. Tech. Rep. 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. Journal of Theoretical Biology 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. Electronic Transactions in Artificial Intelligence 6(12) (2001)
5. Calzone, L., Chabrier-Rivier, N., Fages, F., Soliman, S.: Machine learning biochemical networks from temporal logic properties. In: Plotkin, G. (ed.) Transactions on Computational Systems Biology VI, Lecture Notes in BioInformatics, vol. 4220, pp. 68–94. Springer-Verlag (Nov 2006), cMSB'05 Special Issue
6. Deng, K., Bourke, C., Scott, S.D., Sunderman, J., Zheng, Y.: Bandit-based algorithms for budgeted learning. In: ICDM (2007)
7. Deng, K., Zheng, Y., Bourke, C., Scott, S., Masciale, J.: New algorithms for budgeted learning. Mach Learn 90 (2013)
8. Fages, F., Martinez, T., Rosenblueth, D., Soliman, S.: Influence systems vs reaction systems. In: E. Bartocci, P. Lio, N.P. (ed.) CMSB'16: Proceedings of the fourteenth international conference on Computational Methods in Systems Biology. Lecture Notes in BioInformatics, vol. 9859, pp. 98–115. Springer-Verlag (Sep 2016)
9. Fages, F., Soliman, S.: Abstract interpretation and types for systems biology. Theoretical Computer Science 403(1), 52–70 (2008),
10. Gebser, M., Kaufmann, B., Neumann, A., Schaub, T.: clasp: A conflict-driven answer set solver. In: In Proc. LPNMR'07. pp. 260–265. Springer (2007)
11. Gebser, M., Schaub, T., Thiele, S., Usadel, B., Veber, P.: Detecting inconsistencies in large biological networks with answer set programming. In: de la Banda, M.G., Pontelli, E. (eds.) ICLP'08, Proceedings of the 24th International Conference on Logic Programming. Lecture Notes in Computer Science, vol. 5366, pp. 130–144. Springer-Verlag (2008)
12. Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. Journal of Physical Chemistry 81(25), 2340–2361 (1977)
13. Gordon, A.D., Henzinger, T.A., Nori, A.V., Rajamani, S.K.: Probabilistic programming. In: Proceedings of the on Future of Software Engineering. pp. 167–181. FOSE 2014, ACM, New York, NY, USA (2014)
14. Llamosi, A., Mezine, A., d'Alché 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.) Machine Learning and Knowledge Discovery in Databases ECML PKKDD'14. Lecture Notes in Artificial Intelligence, vol. 8724, pp. 306–321. Springer-Verlag (Sep 2014)
15. Mendoza, L.: A network model for the control of the differentiation process in Th cells. Biosystems 84(2), 101–114 (2006)
16. 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 Systems Biology 8(1), 1–18 (2014)
17. Muggleton, S.H.: Inverse entailment and progol. New Generation Computing 13, 245–286 (1995)
18. 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)
19. 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, pp. 56–72. Springer-Verlag, Berlin, Heidelberg (2006)
20. Thomas, R.: Boolean formalisation of genetic control circuits. Journal of Theoretical Biology 42, 565–583 (1973)
21. Thomas, R.: Regulatory networks seen as asynchronous automata : a logical description. Journal of Theoretical Biology 153, 1–23 (1991)
22. Valiant, L.: A theory of the learnable. Communications of the ACM 27(11), 1134–1142 (1984)
23. Valiant, L.: Probably Approximately Correct. Basic Books (2013)
24. Videla, S., Konokotina, I., Alexopoulos, L.G., Saez-Rodriguez, J., Schaub, T., Siegel, A., Guziolowski, C.: Designing experiments to discriminate families of logic models. Frontiers in Bioengineering and Biotechnology 3, 131 (2015)

Appendix

```
{"IL12" : 0, "IL4": 0}
-> IL12 (0.01)
-< IL12 (0.01)
-> IL4 (0.01)
-< IL4 (0.01)
STAT4 -> IFNg
TBet -> IFNg
/ STAT4,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 6: Code for the lymphocyte differentiation of example 5.

Function	Number of samples
GATA3-	360
GATA3+	361
STAT6-	546
STAT6+	547
STAT4+	805
STAT4-	805
IL12R-	849
IL12R+	849
IL4R-	862
IL4R+	863
IL12+	1096
IL12-	1096
IL4-	2204
IL4+	2205
TBet-	7135
TBet+	7135
STAT1+	9359
STAT1-	9359
IFNgR+	12026
IFNgR-	12026
SOCS1+	12600
SOCS1-	12600
IFNg-	25103
IFNg+	25103

Fig. 4. Repartition of samples in a simulation of the lymphocyte influence system for a single trace. Coefficient of variation is around 1.2.

Function	Number of samples
GATA3+	3861
GATA3-	3869
STAT6+	5614
STAT6-	5621
STAT4+	7980
STAT4-	8010
IL12R+	8420
IL12R-	8446
IL4R+	8665
IL4R-	8671
IL12-	10651
IL12+	10651
IL4-	21888
IL4+	21889
TBet+	69134
TBet-	69164
STAT1+	91070
STAT1-	91114
IFNgR+	116681
IFNgR-	116721
SOCS1+	122546
SOCS1-	122589
IFNg+	242331
IFNg-	242357

Fig. 5. Repartition of samples in a simulation of the lymphocyte influence system for a hundred traces with random starting points. Coefficient of variation is still around 1.2.

Function	No random restart	Random restart
IL12+	0.7	0.7
IL12-	1.1	1.0
IL4+	0.8	0.9
IL4-	1.5	1.5
STAT4+	0.6	0.7
STAT4-	3.6	3.6
IFNg+	0.1	0.1
IFNg-	0.2	0.2
TBet+	0.2	0.2
TBet-	0.2	0.2
GATA3+	3.2	3.3
GATA3-	2.8	2.9
STAT1+	0.2	0.2
STAT1-	0.2	0.2
IFNgR+	0.1	0.1
IFNgR-	0.2	0.3
SOCS1+	0.2	0.3
SOCS1-	0.2	0.2
IL4R+	1.0	1.1
IL4R-	2.9	3.1
IL12R+	1.3	1.2
IL12R-	2.8	2.7
STAT6+	1.9	2.1
STAT6-	3.3	3.7

Fig. 6. Percentages of samples that have never been seen before, for each function.

```
IL12+ : !IL12 /\ (!TBet \/ !GATA3 ) /\ (!TBet \/ !IL4R ) /\
    (!TBet \/ !STAT6 )
IL12- : IL12 /\ (!TBet \/ !GATA3 )
IL4+ : !IL4
IL4- : IL4
STAT4+ : !STAT4 /\ !GATA3 /\ IL12R
STAT4- : STAT4 /\ (GATA3 \/ !IL12R )
IFNg+ : !IFNg /\ (STAT4 \/ TBet )
IFNg- : IFNg /\ (!STAT4 \/ !TBet )
TBet+ : !TBet /\ !GATA3 /\ STAT1
TBet- : TBet /\ (GATA3 \/ !STAT1 )
GATA3+ : !TBet /\ !GATA3 /\ STAT6
GATA3- : GATA3 /\ (TBet \/ !STAT6 )
STAT1+ : !STAT1 /\ IFNgR
STAT1- : STAT1 /\ !IFNgR
IFNgR+ : IFNg /\ !IFNgR /\ !SOCS1
IFNgR- : IFNgR /\ (!IFNg \/ SOCS1 )
SOCS1+ : !SOCS1 /\ (TBet \/ STAT1 )
SOCS1- : SOCS1 /\ (!TBet \/ !STAT1 )
IL4R+ : IL4 /\ !SOCS1 /\ !IL4R
IL4R- : IL4R /\ (!IL4 \/ SOCS1 )
IL12R+ : IL12 /\ !IL12R /\ !STAT6 /\ (!TBet \/ !GATA3 )
IL12R- : IL12R /\ (!IL12 \/ STAT6 )
STAT6+ : IL4R /\ !STAT6 /\ (!TBet \/ !GATA3 )
STAT6- : !IL4R /\ STAT6 /\ (!TBet \/ !GATA3 )
```

Listing 7: Results of PAC learning on 140 simulation of the lymphocyte influence system, with random starting points.

```
{
"IFNg": ["STAT4","TBet"],
"IL4": ["GATA3","STAT1"],
"IL12" : [],
"IFNgR": ["IFNg", "SOCS1"],
"IL4R": ["IL4","SOCS1"],
"IL12R": ["IL12", "STAT6"],
"STAT1":  ["IFNgR"],
"STAT6": ["IL4R"],
"STAT4": ["IL12R","GATA3"],
"SOCS1":  ["STAT1","TBet"],
"TBet": ["STAT1","TBet","GATA3"],
"GATA3": ["STAT6","TBet"]
}
```

Listing 8: Prior knowledge for the lymphocyte model. For each species, a set of possible influencers is given. The PAC algorithm will then learn a model in which only the specified influencers can either induce or inhibit the species.

```
IL12+ : True
IL12- : True
IL4+ : True
IL4- : True
STAT4+ : !GATA3 /\ IL12R
STAT4- : (GATA3 \/ !IL12R )
IFNg+ : (STAT4 \/ TBet )
IFNg- : (!STAT4 \/ !TBet )
TBet+ : !TBet /\ !GATA3 /\ STAT1
TBet- : TBet /\ !GATA3 /\ !STAT1
GATA3+ : !TBet /\ STAT6
GATA3- : !TBet /\ !STAT6
STAT1+ : IFNgR
STAT1- : !IFNgR
IFNgR+ : IFNg /\ !SOCS1
IFNgR- : (!IFNg \/ SOCS1 )
SOCS1+ : (TBet \/ STAT1 )
SOCS1- : (!TBet \/ !STAT1 )
IL4R+ : IL4 /\ !SOCS1
IL4R- : (!IL4 \/ SOCS1 )
IL12R+ : IL12 /\ !STAT6
IL12R- : (!IL12 \/ STAT6 )
STAT6+ : IL4R
STAT6- : !IL4R
```

Listing 9: Results for lymphocyte model, with prior knowledge on the unlabeled influence graph.