Learning Stochastic Finite Automata*

Colin de la Higuera[1] and Jose Oncina[2]

[1] EURISE, Université de Saint-Etienne, 23 rue du Docteur Paul Michelon, 42023 Saint-Etienne, France
cdlh@univ-st-etienne.fr,
[2] Departamento de Lenguajes y Sistemas Informaticos, Universidad de Alicante, Alicante, Spain
oncina@dlsi.ua.es

Abstract. Stochastic deterministic finite automata have been introduced and are used in a variety of settings. We report here a number of results concerning the learnability of these finite state machines. In the setting of identification in the limit with probability one, we prove that stochastic deterministic finite automata cannot be identified from only a polynomial quantity of data. If concerned with approximation results, they become PAC-learnable if the L_∞ norm is used. We also investigate queries that are sufficient for the class to be learnable.

1 Introduction

Probabilistic finite state automata [Paz71] have been introduced to describe distributions over strings. They have been successfully used in several fields, including pattern recognition [LVA+94], computational biology [LPN99] or linguistics [Moh97].

Learning stochastic finite state automata has been an important issue in grammatical inference [KMR+94], with several algorithms already developed and used [CO94b,RST95].Even though the subject has received increasing attention over the past years a systematic study of the different paradigms of learning in this context, with the land stone results, was still missing. It is the ambition of this paper to contribute to this issue.

In section 2 we give the main notations used in this paper: These concern stochastic finite automata and distances. We then visit the different learning paradigms. A first result (section 3) concerns identification in the limit with probability one: We argue that if the definition only requires that the algorithm runs in polynomial update time then even a simple enumerative algorithm is sufficient. On the other hand, if we want identification to be achieved in polynomial time, this is impossible due to the fact that probabilities need to be encoded in some binary scheme.

* This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

We then explore the approximation setting (section 4) and recall that if the distance is computed following the L_∞ norm, then even a simple algorithm that just returns the learning sample is going to PAC-learn automata or any other recursively enumerable family of distributions. Another sort of results (section 5) concerns learning deterministic stochastic finite automata with queries: We introduce various new types of queries and examine learning results with these queries. We conclude in section 6.

2 Notations and definitions

Let Σ be a finite alphabet and Σ^* the (infinite) set of all strings that can be built from Σ. λ will denote the empty string.

A *language* is a subset of Σ^*. By convention, symbols from Σ will be denoted by letters from the beginning of the alphabet $(a, b, c, ...)$ and strings from Σ^* will be denoted by the end of alphabet letters $(..., x, y, z)$. The size of a string $x \in \Sigma^*$ is written $|x|$. Given a string x, $\text{Pref}(x) = \{u : uv = x\}$ and $\text{Suf}(x) = \{v : uv = x\}$. Pref and Suf are extended to sets as $\text{Pref}(X) = \cup_{x \in X} \text{Pref}(x)$ and $\text{Suf}(X) = \cup_{x \in X} \text{Suf}(x)$.

Let $\equiv$ denote an equivalence relation on a set X, then by $[x]$ we are going to denote the class of equivalence of the element x, that is $[x] = \{y \in X : y \equiv x\}$. A *stochastic language* L is a probability distribution over Σ^*. The probability of a string $x \in \Sigma^*$ under the distribution L is denoted as $p_L(x)$. If the distribution is modelled by some syntactic machine A, the probability of x according to the probability distribution defined by A is denoted $p_A(x)$.

We denote by log the base 2 logarithm.

From formal language theory [Har78] we introduce deterministic finite automata (DFA):

Definition 1. *A Deterministic Finite Automaton (*DFA*) A is a tuple $(Q, \Sigma, \delta, q_0, F)$, where: Q is a finite set of* states*; q_0 is the* initial state*; Σ is the alphabet; $\delta : Q \times \Sigma \to Q$ is a* transition function*; $F \subset Q$ is the set of final states.*

We extend in the usual way [Har78] the transition function to a function $\delta : Q \times \Sigma^* \to Q : \delta(q, \lambda) = q, \delta(q, aw) = \delta(\delta(q, a), w)$.

We now adapt this definition to the stochastic setting:

Definition 2. *A Stochastic Deterministic Finite Automaton (*SDFA*) A is a tuple $(Q, q_0, \Sigma, \delta, \pi)$, where Q, Σ, δ, q_0 are as in* DFA *and π is a function with two profiles: $\pi : Q \times \Sigma \to \mathbb{Q}^+$ (*transition probabilities*) and $\pi : Q \to \mathbb{Q}^+$ (*final-state probabilities*).*

Function π is recursively extended to $\pi : Q \times \Sigma^* \to \mathbb{Q}^+$ such that $\pi(q, \lambda) = 1$ and $\pi(q, ax) = \pi(q, a)\pi(\delta(q, a), x)$.

The probability of a string x starting from the state q is defined as $p(q, x) = \pi(q, x)\pi(\delta(q, x))$. The probability of a string x is $p(x) = p(q_0, x)$. Let X be a set of strings, $p(X) = \sum_{x \in X} p(x)$.

We are going to say that a probability distribution L is a *Stochastic Deterministic Regular Language* (SDRL) if it is produced by a SDFA.

As the stochastic languages define probability distributions over Σ^*, it is required that $0 \leq p(x) \leq 1\ \forall x$ and $p(\Sigma^*) = 1$ (*consistence condition*).

Usually $\sum_{a\in\Sigma} \pi(q,a) + \pi(q) = 1, \forall q \in Q$ is also required. This condition is not strictly necessary, as the SDFA $(\{q_0,q_1\}, q_0, \{a\}, \{\delta(q_0,a) = q_1, \delta(q_1,a) = q_0\}, \{\pi(q_0,a) = 27, \pi(q_1,a) = 0.5/27, \pi(q_0) = 0.5, \pi(q_1) = 0\})$ is consistent and does not fulfil the condition. In theorem 5 we describe a method to transform a SDFA that does not fulfil the condition into one that does.

For combinatorial purposes such as needing to compute the complexity of an algorithm, the size of a SDFA must be polynomially linked with the number of bits needed to encode it in a reasonable way. This only makes sense if the probabilities themselves can be encoded; Therefore rational probabilities will be used. In the case of SDFA the number n of states, the size $|\Sigma|$ of the alphabet and the number of bits needed to encode the probabilities are needed. We denote by $\|A\|$ the size of a SDFA A.

When concerned with measuring how close one distribution can be from another, or from a sample, it will be of interest to compare these through distances:

$$d_n(\mathcal{D},\mathcal{D}') = \sqrt[n]{\sum_{w\in\Sigma^*} |p_{\mathcal{D}}(w) - p_{\mathcal{D}'}(w)|^n}$$

$$d_\infty(\mathcal{D},\mathcal{D}') = \max_{w\in\Sigma^*} |p_{\mathcal{D}}(w) - p_{\mathcal{D}'}(w)|$$

This last distance allows favourable convergence bounds, through the use of the following lemma due to Angluin [Ang88]:

Lemma 1. *Let $\mathcal{D}$ be any distribution on Σ^*, let $a > 1$ and let $I(n) = \sqrt{\frac{6a(\log n)}{n}}$. Then,*

- *with probability at least $1 - n^{-a}$,*
- *with probability 1 and for all but a finite number of values of n,*

$$d_\infty(\mathcal{D}, \mathcal{D}\langle n\rangle) \leq I(n)$$

where $\mathcal{D}\langle n\rangle$ is the empirical distribution built from a sample of size n.

3 About polynomial identification in the limit with probability one

Even if grammatical inference is usually considered to be dealing with learning languages, it is clearer to define things in terms of learning grammars or automata. In this paper, we are concerned with automata, and therefore all definitions shall be given in this setting.

We consider a class of languages $\mathcal{L}$ and a class $\mathcal{R}$ of representations for these languages. We are going to denote by $r_{\mathcal{R}}(L)$ (and simplify to $r(L)$ in non

ambiguous contexts) the smallest representation in $\mathcal{R}$ of a language L and by $\|r_{\mathcal{R}}(L)\|$ its size. $L(r)$ denotes the language represented by the description r.

In order to learn we are going to suppose that the data to identify a stochastic language is obtained following the probability distribution and that successive draws are independent. The probability of a sequence of draws is the product of the probabilities of each draw.

Definition 3 (Identification in the limit with probability 1). *A class of stochastic languages $\mathcal{L}$ is learnable*[1] *with probability 1 in terms of a representation class $\mathcal{R}$ if there exists an algorithm $\phi(\cdot)$ that given any language L in $\mathcal{L}$, for any increasing sequence $\langle s_n \rangle$ drawn according to L, $\phi(s)$ returns some representation r. Moreover, with probability 1, for all but a finite number of values of n, $\phi(s)$ returns r such that $L(r) = L$.*

The previous definition has led to the implicit definition of polynomial identification that has been mainly used in the literature [CO94b]. It basically states that identification has to be obtained by using an algorithm that has polynomial runtime. This does not mean that a polynomial number of examples may be sufficient, as discussed in [dlHO03]. We will refer to this definition as weak polynomial identification:

Definition 4 (Weak polynomial identification in the limit with probability 1). *A class of stochastic languages $\mathcal{L}$ is weak polynomially learnable with probability 1 in terms of a representation class $\mathcal{R}$ if there is an algorithm $\phi(\cdot)$ and a polynomial p for which for any language L in $\mathcal{L}$, and any increasing sequence $\langle s_n \rangle$ of strings drawn according to L:*

1. *$\phi(\cdot)$ identifies L in the limit with probability 1;*
2. *$\phi(\cdot)$ works in time in $p(\|\langle s_n \rangle\|)$.*

Under this criterion it has been established that algorithm ALERGIA [CO94b] could learn SDFA, but also that stochastic deterministic linear grammars [dlHO03] could be identified.

Nevertheless this definition has a serious flaw:

Theorem 1. *Let $\mathcal{L}$ be a class of recursivelly enumerable languages that admits a representation class $\mathcal{R}$ for which distance d_∞ is computable. $\mathcal{L}$ when represented by $\mathcal{R}$ can be weakly polynomially learned with probability 1.*

Proof. We follow in this proof arguments from [Pit89] and [Ang88] and use a simple enumerative algorithm that would find the first language consistent (in the sense of lemma 1) in the enumeration identifies in the limit. To make the algorithm fit with the complexity constraints, we just make the algorithm compute the time it is entitled to from the current examples. The algorithm then computes as far as it can go with that time and returns whatever solution it has reached at that point. There is a point where the algorithm will converge. □

[1] We will write learnable for identifiable in the limit.

Corollary 1. SDRL *when represented by* SDFA *can be weakly polynomially identified in the limit with probability 1.*

An alternative definition which would bound the overall time is as follows:

Definition 5 (Strong polynomial identification in the limit with probability 1). *A class of stochastic languages $\mathcal{L}$ is strongly polynomially learnable with probability 1 in terms of a representation class $\mathcal{R}$ if there is an algorithm $\phi(\cdot)$ and two polynomials p and q for which for any language L in $\mathcal{L}$ and any $\delta > 0$, and any increasing sequence $\langle s_n \rangle$ of strings drawn according to L:*

1. *$\phi(\cdot)$ identifies L in the limit with probability 1;*
2. *$\phi(\cdot)$ works in time in $p(\|\langle s_n \rangle\|, \frac{1}{\delta})$;*
3. *If $n \geq q(\|r(L)\|, \frac{1}{\delta})$, $\phi(\cdot)$ computes with probability at least $1 - \delta$ a representation h such that $L(h) = L$.*

The above definition takes into account three aspects: As before, the algorithm is required to identify in the limit and to work in polynomial time; Furthermore, with high probability, identification is expected from a polynomial number of examples only.

Because probabilities can be encoded in very different ways it may seem that results will depend on the way they are encoded. Nevertheless a reasonable encoding of the probabilities means that the size of the encodings will be logarithmic in the number of different probabilities that need encoding.

We prove here that even in the case where we have only to choose one probability out of n, identification requires a number of strings that is too high. To do so we do not make any assumption on the way the learning algorithm is to use the information it receives. To do so we consider n languages L_1, L_2, .., L_n such that they just have 2 strings that are different, x and y: namely $p_{L_i}(x) = p_i$ and $p_{L_i}(y) = 1 - p_i$. We have also $\forall i, j \leq n, p_i \neq p_j$.

Proposition 1. *Let $\mathcal{L}$ be a class of languages that contains $\{L_1, L_2, .., L_n\}$ as above, let $\mathcal{R}$ be any representation class for $\mathcal{L}$, let $\phi(\cdot)$ be any learning algorithm for the class $\mathcal{L}$ in terms of the representation class $\mathcal{R}$ and m be an integer. Then there is an L_i in $\mathcal{L}$ such that for any sequence s of size at most m, $p_{L_i}(\phi(s) = L_i) \leq \frac{(m+1)(m+2)}{2n}$.*

Proof. Let us suppose (for contradiction) that there exists such an algorithm $\phi(\cdot)$.

Consider the sets S_j of all possible sequences of length m over strings $\{x, y\}$. With each sequence s we associate value $v(s) = |s|_x$, the number of times string x appears in the sequence. The set of sequences of S_j with value $v(s) = i$ will be denoted S_j^i.

Lemma 2. *Let $m > 0$, there is a language L in $\mathcal{L}$ such that for each $j : 0 \leq j \leq m$ and each $i : 0 \leq i \leq j$*

$$\frac{|\{s \in S_j^i : \phi(s) = L\}|}{|S_j^i|} = \frac{|\{s : \phi(s) = L \wedge |s| = j \wedge v(s) = i\}|}{|\{s : |s| = j \wedge v(s) = i\}|} \leq \frac{(m+1)(m+2)}{2n}$$

As the sequences s of each S_j^i have all the same probability, using lemma 2 for such a language L the sum of all probabilities of sequences drawn according to L is less than $\frac{(m+1)(m+2)}{2n}$. □

Proof (of lemma 2). Suppose this is not true. Then for each of the n languages L in $\mathcal{L}$ there are $i, j : 0 \leq i \leq j \leq m$ with more than $\frac{(m+1)(m+2)}{2n}$ sequences for which $\phi(s) = L$. But such associations can only be produced less than $\frac{(m+1)(m+2)}{2n} \cdot n$ times. Hence this cannot be true for each language L. □

If now we restrict ourselves to reasonable representations we have:

Corollary 2. *Let $\mathcal{L}$ be a class of languages that contains $\{L_1, L_2, .., L_n\}$ as above, let $\mathcal{R}$ be any representation class for $\mathcal{L}$ such that $\|r(L_i)\| < p(\log n)$ for some polynomial $p(\cdot)$, let $\phi(\cdot)$ be any learning algorithm for the class $\mathcal{L}$ in terms of the representation class $\mathcal{R}$ and let $q(\cdot)$ be a polynomial. Then there is an L_i and a constant k such that for any sequence of size at most $q(\|r(L_i)\|)$, $p_{L_i}(\phi(s) = L_i) \leq \frac{\log^{2k} n}{n}$.*

Proof. As $\|r(L_i)\| < p(\log n)$ and $q(\cdot)$ is a polynomial it is enough to show the existence of L_i for all sequences of length at most $m < \log^k n$ for any fixed k. □

As this applies to regular languages when represented by SDFA as a corollary we have:

Theorem 2. *The class of* SDRL *is not strongly polynomially learnable with probability 1 by means of* SDFA.

4 Polynomial approximation of languages

To be able to identify both the structure and the individual probabilities have to be exactly identified. This may legitimately prove to be a too hard task. An alternative is then to adapt Valiant's PAC-learning model [Val84] to the stochastic setting. We base ourselves here on the definitions from [RST95]:

Definition 6 (ϵ-good hypothesis). *Let A be the target* SDFA *and B be a hypothesis* SDFA. *We say that B is an ϵ-good hypothesis with respect to A for $\epsilon \geq 0$ if $d(L(A), L(B)) < \epsilon$.*

A learning algorithm is now asked to learn a grammar given a *confidence* parameter δ and an *approximation* parameter ϵ. The algorithm can also be given an upper bound on the size of the target grammar and has access to a learning sample.

Definition 7 (Polynomial Pac-learnable). *A class of stochastic languages $\mathcal{L}$ is* polynomially PAC-learnable *in terms of a representation class $\mathcal{R}$ (using distance d) if there exists an algorithm $\phi(\cdot)$ and a polynomial $p()$ such that for any language L in $\mathcal{L}$ if ϕ is given a learning sample s of size $p(\|r(L)\|)$ it returns a representation r such that $L(r)$ is an ϵ-good hypothesis with respect to L with probability at least 1-δ in time polynomial in $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, $|\Sigma|$, $\|r(L)\|$ and the length of the longest example in s.*

This definition depends strongly on the distance measure that is chosen:

Proposition 2. SDRL *when represented by* SDFA *is polynomially* PAC*-learnable using distance* d_∞.

Proof. This is a consequence of lemma 1. Let us consider a simple algorithm that we will call RETURNPTA which given a learning sample constructs a SDFA that represents exactly the empirical distribution given by the sample. Obviously RETURNPTA is no good at identifying SDFA. Nevertheless RETURNPTA will polynomially PAC-learn SDFA as with high probability the distance according to norm L_∞ between the empirical distribution and the target distribution converges very fast. □

On the other hand, it can easily be shown that the above result is no longer true when the distance is taken according to another norm. For instance for the norms L_1 and L_2, a language which shares the mass of probabilities in a uniform way over an exponential number of strings will not be closely approximated by the empirical distribution drawn from only a polynomial number of strings.

5 Polynomial learning with queries

Queries have been used in order to provide an alternative learning setting, a more favourable one, but also one that can be better controlled. Typical queries include equivalence and membership queries [Ang87] and extended membership queries [BV96]. An *extended membership query* is built from a string x. The oracle has to return the probability of x. An *equivalence query* is built from a represetation r. The oracle has to return "yes" if $L(r)$ is the target language or a counterexample if not. Angluin [Ang87] gives the following definition of learning with queries:

Definition 8. *A class of grammars is learnable with queries of type T if*

1. *It works in polynomial update time (in the size of the target and of the longest counter-example);*
2. *The number of queries always is polynomial (in the size of the target).*

5.1 Extended membership queries

It is easy to show that SDRL (represented by SDFA) are learnable from extended membership queries only if regular languages (represented by DFA) are learnable from membership queries, which is known not to be the case [Ang81].

Theorem 3. *The class of* SDRL *when represented by* SDFA *is not learnable from extended membership queries only.*

Proof. If not DFA would be identifiable from membership queries. We construct from a completed DFA $A = (Q, \Sigma, \delta, q_0, F)$ a SDFA $B = (Q', \Sigma', \delta', q'_0, \pi)$ as follows:

- $Q' = Q \cup \{q_f\}$; $\Sigma' = \Sigma \cup \{+, -\}$; $q'_0 = q_0$;
- $\forall q \in Q$, $\delta'(q, +) = q_f$, $\delta'(q, -) = q_f$; $\forall q \in Q, \forall a \in \Sigma$, $\delta'(q, a) = \delta(q, a)$;
- $\forall q \in Q, \forall a \in \Sigma, \pi(q) = 0, \pi(q_f) = 1, \pi(q, a) = \frac{1}{2 \cdot |\Sigma|}, \pi(q_f, a) = 0$
- $\forall q \in Q, q \in F \Rightarrow \pi(q, +) = \frac{1}{2}, \pi(q, -) = 0$; $q \notin F \Rightarrow \pi(q, +) = 0, \pi(q, -) = \frac{1}{2}$

The above construction is made from a completed DFA, *i.e.* a DFA to which eventually an extra non final state has been added and is reached by all absent transitions. This ensures that through the construction we have $w \in L_A \Leftrightarrow p_B(w+) = \frac{(2 \cdot |\Sigma|)^{-|w|}}{2}$ and $w \notin L_A \Leftrightarrow p_B(w-) = \frac{(2 \cdot |\Sigma|)^{-|w|}}{2}$. An extended membership query therefore gives us the same information on the underlying SDFA as a membership query would. □

The reasons explaining the above result are that some transition, or some state in the automaton can be barely reachable. *Lock automata*, introduced by Angluin [Ang81] in the DFA setting, are hard to learn. In these automata one long string is going to be the *lock* to the automaton, and only by guessing this string is learning possible.

5.2 Extended membership queries and equivalence queries

To avoid the problem due to lock automata there are usually three possibilities:

- Accept to make a small (ϵ) error and therefore to learn in a PAC setting;
- Give the algorithm a specific learning set (called a *teaching set* or *complete set*) of strings indicating that certain strings will have to be specifically checked;
- Introduce equivalence queries.

Carrasco and Oncina [CO99] showed that given a SDRL L, the minimal SDFA A such that $L(A) = L$ fulfils that

$$\delta(q_0, x) = \delta(q_0, y) \iff \frac{p(xv)}{p(x\Sigma^*)} = \frac{p(yv)}{p(y\Sigma^*)} \; \forall v \in \Sigma^*$$

Then, if two strings (x, y) reach two non equivalent states, there are two strings (u,v) such that $\frac{p(xu)}{p(xv)} \neq \frac{p(yu)}{p(yv)}$. Then, the following definition makes sense:

Definition 9. *A set of examples I is* complete *if every probability in the automaton is exercised and if all pairs of states can be separated:*

1. $\forall q, q' \in Q, \forall a \in \Sigma : \delta(q, a) = q', \exists uav \in I : \delta(q_0, u) = q$.
2. $\forall q, q' \in Q, q \neq q', \exists xu, xv, yu, yv \in I :$
   $\delta(q_0, x) = q, \delta(q_0, y) = q', \frac{p(xu)}{p(xv)} \neq \frac{p(yu)}{p(yv)}$

Theorem 4. *The class of* SDRL *when represented by* SDFA *is learnable from extended membership queries and a complete set.*

Proof. The algorithm is based on the construction of a 2 entries table (T) indexed respectively by the prefixes and the suffixes of the learning set I. Each cell of the table is filled by making an extended membership query, with the probability of the concatenation of its indexes. That is, $T(x, y) = p_T(xy)$.

Let us define the following equivalence relation:

$$x \equiv y \iff \exists \alpha > 0 : \forall w \in \mathrm{Suf}(I)\ T(x, w) = \alpha T(y, w)$$

From the table, the structure of an automaton A is built: $Q = \{[x] : x \in \mathrm{Pref}(I)\}$, $q_0 = [\lambda]$, $\delta([x], a) = [xa]$.

By construction of the complete set, all the states of the target automaton can be reached by a prefix of a string in I, and for all pairs of non equivalent states in the target automaton we know that there exists a pair of strings that makes the equivalence false. Obviously, if two states are equivalent in the target automaton, they are also equivalent in the hypothesis automaton.

Then we have the structure of the hypothesis automaton. Now the remaining problem is how to assign the probabilities of the hypothesis automaton. We are going to follow some ideas from [CP71].

On the following, subscript T will denote items related to the target autonaton, while subscript A will denote items related to the hipothesis automaton.

Let w_i be a string such that $\delta(q_0, w_i) = q_i$ (we force $w_0 = \lambda$) and let v_i be such that $p_T(w_i v_i) \neq 0$.

We are going to assign

$$\pi_A(q_i) = p_T(w_i); \qquad \pi_A(q_i, a) = \frac{p_T(w_i a v_j)}{p_T(w_j v_j)} \qquad \text{where } q_j = \delta(q_i, a)$$

This automaton does not fulfil the condition $\pi_A(q_i) + \sum_{a\in\Sigma} \pi_A(q_i, a) = 1$ but we are going to show that $p_A(w) = p_T(w)\ \forall w$.

As $p_T(w_i v_i) = \pi_T(q_0, w_i) p_T(q_i, v_i)$, and let $q_j = \delta(q_i, w)$, we can write

$$\pi_A(q_i, a) = \frac{\pi_T(q_0, w_i)}{\pi_T(q_0, w_j)} \pi_T(q_i, a); \qquad \pi_A(q_i, w) = \frac{\pi_T(q_0, w_i)}{\pi_T(q_0, w_j)} \pi_T(q_i, w)$$

Let now $q_i = \delta(q_0, w)$. Then $p_A(w) = \pi_A(q_0, w)\pi_A(q_i)$
$= \frac{\pi_T(q_0,\lambda)}{\pi_T(q_0,w_i)} \pi_T(q_0, w) p_T(w_i) = \pi_T(q_0, \lambda)\pi_T(q_0, w)\pi_T(q_i) = p_T(w)$

□

Theorem 5. *There is an algorithm that given any* SDFA A *constructs a* SDFA B *such that* $L(A) = L(B)$ *and* $\pi_B(q_i) + \sum_{a\in\Sigma} \pi_B(q_i, a) = 1\ \forall q_i \in Q_B$.

Proof. From SDFA A define a row vector F, a column vector S, and a matrix M_a for each $a \in \Sigma$ as:

$$S = (1, 0, \ldots, 0); \quad F_i = \pi_A(q_i); \quad (M_a)_{i,j} = \begin{cases} \pi(q_i, a) & \text{if } q_j = \delta(q_i, a) \\ 0 & \text{otherwise} \end{cases}$$

Then, the probability of a string $a_1 \dots a_n$ can be computed as $p(a_1 \dots a_n) = SM_{a_1} \dots M_{a_n} F$. Let $M_{a_1 \dots a_n} = M_{a_1} \dots M_{a_n}$ and let $M = \sum_{a \in \Sigma} M_a$.

Let us define the automaton B with the same structure that A but with the function π defined as follows:

$$\pi_B(q_i) = \frac{p_A(w_i)}{p_A(w_i\Sigma^*)}; \qquad \pi_B(q_i, a) = \frac{p_A(w_i a\Sigma^*)}{p_A(w_i\Sigma^*)}$$

where w_i are arbitrary strings such that $\delta(q_0, w_i) = q_i$. Note that $p(w\Sigma^*) = SM_w \sum_{i=0}^{\infty} M^i F = SM_w(I - M)^{-1}F$ where I is the unity matrix.

It is easy to verify that B fulfils the normalisation condition. It remains to be shown that $\forall x, p_A(x) = p_B(x)$. First note that, $\forall x, y : \delta(q_0, x) = \delta(q_0, y) = q_i$,

$$\frac{p(xav)}{p(xv)} = \frac{\pi(q_0, x)p(q_i, av)}{\pi(q_0, x)p(q_i, v)} = \frac{\pi(q_0, y)p(q_i, av)}{\pi(q_0, y)p(q_i, v)} = \frac{p(yav)}{p(yv)}$$

then $\frac{Ap(xa\Sigma^*)}{p(x\Sigma^*)} = \frac{p(ya\Sigma^*)}{p(y\Sigma^*)}$ and similarly $\frac{p(x)}{p(x\Sigma^*)} = \frac{p(y)}{p(y\Sigma^*)}$.

Let us now consider a string $a_1 \dots a_n$, let we call $q_i = \delta(q_0, a_1 \dots a_i)$, then

$$p_B(a_1 \dots a_n) = \frac{p_A(w_0 a_1\Sigma^*)}{p_A(w_0\Sigma^*)} \frac{p_A(w_1 a_2\Sigma^*)}{p_A(w_1\Sigma^*)} \dots \frac{p_A(w_{n-1} a_n\Sigma^*)}{p_A(w_{n-1}\Sigma^*)} \frac{p_A(w_n)}{p_A(w_n\Sigma^*)} =$$
$$\frac{p_A(a_1\Sigma^*)}{p_A(\Sigma^*)} \frac{p_A(a_1 a_2\Sigma^*)}{p_A(a_1\Sigma^*)} \dots \frac{p_A(a_1 \dots a_{n-1} a_n\Sigma^*)}{p_A(a_1 \dots a_{n-1}\Sigma^*)} \frac{p_A(a_1 \dots a_n)}{p_A(a_1 \dots a_n\Sigma^*)} = p_A(a_1 \dots a_n)$$

□

It is reasonable to use equivalence queries instead of a complete set, as in both cases the construction of the SDFA from the table can be done in the same way. We thereby claim that:

Theorem 6. *The class of* SDRL *when represented by* SDFA *is polynomially learnable from extended membership queries and equivalence queries.*

5.3 Extended prefix language queries

An extended prefix language query is made by submitting to the oracle a string w. The oracle then returns the probability $p(w\Sigma^*)$. It can be noticed that an extended membership query can easily be simulated through $|\Sigma|$ extended prefix language queries.

Theorem 7. *The class of* SDRL *when represented by* SDFA *is not polynomially learnable from extended prefix language queries.*

Proof. Let $w \in \Sigma^n$ and consider the following language L_w: $\forall x \in \Sigma^n, x = w \Rightarrow p(x) = 0, x \neq w \Rightarrow p(x) = \frac{1}{2^n}$, $p(wa) = \frac{1}{2^n}$ This language is recognised by a SDFA with at most $2n + 2$ states. Call L_n the set of all languages L_w with $w \in \Sigma^n$. Now let the oracle answer to each extended prefix language query "x" with the quantity $\frac{1}{2^{|x|}}$ if $|x| \leq n$, 0 if not. Then it is straightforward that in the worse case at least 2^n queries are needed. □

Since extended membership queries can be simulated by extended prefix language queries, it follows that:

Theorem 8. *The class of* SDRL *when represented by* SDFA *is polynomially learnable from extended prefix language queries and equivalence queries.*

6 Open questions and conclusions

From this point, a number of alternative routes are open. We mention two problems and research directions that deserve in our view to be investigated, and conclude.

- Extended membership queries were introduced in [BV96]: The learning algorithm may ask for the value $p(x)$ on strings x of its choice.
  We refine the concept to the case where the answer to the query can be an approximate answer: A *specific sampling* query is made by submitting a pattern: The oracle draws a string matching pattern sampled according to the distribution $\mathcal{D}$. Specific sampling queries are intended to fit the idea that the user can ask for examples matching some pattern he is interested in.
  For example specific sampling $a\Sigma^*b$ requires an example starting with a and ending with b, sampled following the distribution induced by $\mathcal{D}$.
  We conjecture that these queries should be able to help us learn the class of SDFA.
- In the previous sections we have shown that both identification and PAC-learning are hard because of the infinite number of probability values that can be reached. So, we suggest to limit the set of probabilities of the automaton. The idea here is to learn SDFA but where the probabilities are predefined, *i.e.* come from a fixed set. We would typically consider $K-$SDFA built from a set $\mathcal{P}_K = \{\frac{i}{K} : 0 \leq i \leq K\}$ of predefined probabilities, and taking all transition probabilities from $\mathcal{P}_K$.
  We believe that that the identification of the probabilities becomes easier in this setting, and that the class of $K-$SDFA can now be learnable. We conjecture the class to be PAC-learnable using distances d_1 and d_2.

We have proved that when considering identification the problem of learning SDFA was in most cases intractable.

We have suggested 2 options in order to obtain new positive results: The first consisted in allowing the sampling to be directed. We believe that positive PAC-learning results are possible in this case.

A second, and possibly more practical approach, is to severely reduce the class of stochastic languages under consideration, not by taking a smaller support class, but by simplifying the expression of the probabilities. This is a a way of adding bias to the problem and might allow SDFA to be better fitted for language modeling tasks.

Acknowledgement: The authors thank Franck Thollard for pointing out to them the result from section 4.

References

[Ang81] D. Angluin. A note on the number of queries needed to identify regular languages. *Information and Control*, 51:76–87, 1981.

[Ang87] D. Angluin. Queries and concept learning. *Machine Learning Journal*, 2:319–342, 1987.

[Ang88] D. Angluin. Identifying languages from stochastic examples. Technical Report YALEU/DCS/RR-614, Yale University, March 1988.

[BV96] F. Bergadano and S. Varricchio. Learning behaviors of automata from multiplicity and equivalence queries. *SIAM Journal of Computation*, 25(6):1268–1280, 1996.

[CO94a] R. C. Carrasco and J. Oncina, editors. *Grammatical Inference and Applications, Proceedings of ICGI '94*, number 862 in LNAI, Berlin, Heidelberg, 1994. Springer-Verlag.

[CO94b] R. C. Carrasco and J. Oncina. Learning stochastic regular grammars by means of a state merging method. In ICGI'94 [CO94a], pages 139–150.

[CO99] R. C. Carrasco and J. Oncina. Learning deterministic regular grammars from stochastic samples in polynomial time. *RAIRO (Theoretical Informatics and Applications)*, 33(1):1–20, 1999.

[CP71] J. W. Carlyle and A. Paz. Realizations by stochastic finite automata. *Journal of Computation and System Sciences*, (5):26–40, 1971.

[dlHO03] C. de la Higuera and J. Oncina. Identification with probability one of stochastic deterministic linear languages. In *Proceedings of ALT 2003*, LNCS, pages 134–148, Berlin, Heidelberg, 2003. Springer-Verlag.

[Har78] M. H. Harrison. *Introduction to Formal Language Theory*. Addison-Wesley Publishing Company, Inc., Reading, MA, 1978.

[KMR+94] M. J. Kearns, Y. Mansour, D. Ron, R. Rubinfeld, R. E. Schapire, and L. Sellie. On the learnability of discrete distributions. In *Proc. of the 25th Annual ACM Symposium on Theory of Computing*, pages 273–282, 1994.

[LPN99] R. B. Lyngsø, C. N. S. Pedersen, and H. Nielsen. Metrics and similarity measures for hidden Markov models. In *Proceedings of ISMB'99*, pages 178–186, 1999.

[LVA+94] S. Lucas, E. Vidal, A. Amari, S. Hanlon, and J. C. Amengual. A comparison of syntactic and statistical techniques for off-line OCR. In Carrasco and Oncina [CO94a], pages 168–179.

[Moh97] M. Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(3):269–311, 1997.

[Paz71] A. Paz. *Introduction to probabilistic automata*. Academic Press, NY, 1971.

[Pit89] L. Pitt. Inductive inference, DFA's, and computational complexity. In *Analogical and Inductive Inference*, number 397 in LNAI, pages 18–44. Springer-Verlag, Berlin, Heidelberg, 1989.

[RST95] D. Ron, Y. Singer, and N. Tishby. On the learnability and usage of acyclic probabilistic finite automata. In *Proceedings of COLT 1995*, pages 31–40, 1995.

[Val84] L. G. Valiant. A theory of the learnable. *Communications of the Association for Computing Machinery*, 27(11):1134–1142, 1984.