Machine Learning, 44, 37–66, 2001

Learning Regular Languages from Simple Positive Examples

FRANÇOIS DENIS denis@lifl.fr
Equipe Grappa, LIFL, Université de Lille 1, 59655 Villeneuve d'Ascq Cedex, France

Editors: Colin de la Higuera and Vasant Honavar

Abstract. Learning from positive data constitutes an important topic in Grammatical Inference since it is believed that the acquisition of grammar by children only needs syntactically correct (i.e. positive) instances. However, classical learning models provide no way to avoid the problem of overgeneralization. In order to overcome this problem, we use here a learning model from *simple* examples, where the notion of simplicity is defined with the help of Kolmogorov complexity. We show that a general and natural heuristic which allows learning from *simple positive examples* can be developed in this model. Our main result is that *the class of regular languages is probably exactly learnable from simple positive examples.*

Keywords: grammatical inference, PAC learning, Kolmogorov complexity, regular languages, deterministic finite automata

1. Introduction

Natural language learning constitutes one of the most typical human learning abilities. It is also one of the most difficult challenge for researchers in Computational learning theory. In both reference models used in Learning theory, namely Gold's identification in the limit model (Gold, 1967) and Valiant's probably approximatively correct model (Valiant, 1984), results are not satisfactory, even for one of the simplest class of formal languages: the class REG of regular languages. Either REG is not learnable (as in PAC model modulo some usual cryptographic assumption (Kearns and Valiant, 1994)) or REG is learnable with the help of a trivial algorithm which has no cognitive relevance (as in Gold's model (Gold, 1967)). See Pitt (1989), and Sakakibara (1997) for a survey of the field.

Things are even getting worse when we try to take a natural constraint into account: natural language learning is based on sentences which are syntactically correct. Therefore, formal theories must explain how it is possible to learn from positive data only. Gold proves in 1967 that, as soon as a class of languages contains all of the finite languages and at least one infinite language, it is not identifiable in the limit from positive data. As a corollary, the class REG is not learnable from positive data in Gold's model. The problem is that no positive example can refute a too general hypothesis. Therefore, it seems impossible to avoid overgeneralization, except in the simplest cases.

Angluin has given a characterization of indexed families of recursive languages which are identifiable in the limit from positive examples (Angluin, 1980). She gave a gen-

eral heuristic in order to avoid overgeneralization: at each step n, if the current positive sample is S, a hypothesis h may be output if it is consistent with S and *if some computable finite sample* $S_{h,n}$ *of* h *is included in* S. The sets $S_{h,n}$ can be interpreted as *characteristic* samples for h that must be present if h must be output. In other words, a too general hypothesis will be refuted if its characteristic sample is not present in the current sample. Angluin (1982) has introduced subclasses of REG, called k-reversible languages, and shown the existence of polynomially computable characteristic samples sufficient for identification from positive data. See also Sakakibara (1992), Yokomori (1995), de Jongh and Kanazawa (1996), Kanazawa (1996), Koshiba (1997), and Head et al. (1998).

Due to the free distribution and polynomial running time requirements, results are still weaker in PAC learning framework (Natarajan, 1991b; Shvayster, 1990; Yokomori, 1995; Denis, 1998).

Many very valuable heuristics and learning algorithms from positive examples alone have been proposed yet and many of them have been used quite successfully in some practical situations such as speech recognition, and natural language learning. For example, see Carrasco and Oncina (1994) for an approach based on Probabilistic finite state automata (PFSA) and Stolcke and Omohundro (1994), Rabiner and Juang (1986) for a Hidden Markov Model (HMM) approach. But, to our knowledge, no general result is available for PFSA nor for HMMs. Our goal in this paper is to study under what conditions general classes of languages can be learned efficiently from positive examples.

Gold suggests that one reason why natural language learning is possible is that the learner is not provided with *arbitrary* examples (Gold, 1967). There are several ways to give substance to this idea:

– The learner may ask *queries*. Angluin proved in 1987 that REG is exactly learnable within polynomial time using membership and equivalence queries, i.e. using a Minimally Adequate Teacher (MAT). We think that such queries are not meaningful in a positive learning framework. Yet, it is often remarked that children get feedback about what they say. For example, parents commonly repeat what their children say with corrections. But, against these arguments, it can be said that natural language learning mainly develops before children are systematically corrected by their parents. And completely incorrect utterances are rarely observed. We think that membership queries should be restricted in some way in order to be used in a positive learning framework.
– We may impose a *teaching set* to be present in every current sample (Gold, 1978; Angluin, 1987; Goldman & Mathias, 1996). We mainly study here the learning model of Goldman and Mathias. If the teaching set may contain negative examples, the class REG is efficiently exactly learnable in this model (Goldman & Mathias, 1996; Oncina & Garcia, 1992). But if the teaching set must be composed of positive examples only, it is easy to show that REG is not learnable.
– In PAC framework, the class of allowed distributions can be restricted (Li & Vitányi, 1991; Denis, D'Halluin & Gilleron 1996; Denis & Gilleron, 1997a; D'Halluin, 1998). The starting point in Li and Vitányi (1991) is to suppose that *simple* examples must be given more importance in the learning process. Kolmogorov Complexity $K(x)$ is used to measure the complexity of an example x and it is supposed that in the learning

process, the examples are drawn according to the Solomonoff-Levin distribution $\mathbf{m}$ (where $\mathbf{m}(x) \simeq 2^{-K(x)}$). This model has been generalized in Denis et al. (1996) in order to take the complexity of (a representation of) the target c into account: it is supposed that the examples are drawn according to the *conditional* Solomonoff-Levin distribution $\mathbf{m}_c$ (where $\mathbf{m}_c(x) \simeq 2^{-K(x|c)}$). Simple examples (i.e. of low conditional complexity $K(x \mid c)$) are more probable under $\mathbf{m}_c$. A class of concepts is said to be *learnable from simple examples*, or PACS learnable in short, if it is PAC learnable provided the examples are drawn according to $\mathbf{m}_c$. It has been proved in Parekh and Honavar (1997) that the class REG is PACS learnable.

Learning models with teaching sets raise a problem when a representation of the target concept can be encoded by means of examples. We show that in this case, every learning algorithm can be rendered trivial: indeed, nothing prevents the teacher to put an encoding of the target concept into the teaching set and nothing prevents the learner from using it. Unfortunately, the class of regular languages is concerned with such a *collusion phenomenon* between the teacher and the learner, since it is possible to encode efficiently a DFA by a not too large string. As the string which encodes the minimal DFA that recognizes a language L has a low Kolmogorov complexity knowing a representation of L, it is a simple example for L. Therefore, the learning model from simple examples is also concerned with this problem. This means that these models are not strong enough to avoid collusion phenomena and that if we still want to use them, we must verify in each case that such a global information on the target cannot be directly provided to the learner.

Despite the problems pointed out above, we are mainly interested in the learning model from simple examples. There are several reasons for that:

1. The theory of Kolmogorov complexity provides a rigorous way to define the amount of information shared between an example and a target language. If we say that an example is *characteristic* of a language L when it has a small Kolmogorov complexity relatively to a representation of L, we give a rigorous definition that captures important features of this intuitive notion.
2. It seems that there is no way to efficiently encode a regular language by means of positive examples.
3. Supposing that simple examples have a higher weight in the learning process suggests to use the following heuristic to avoid overgeneralization: if h is a hypothesis consistent with the current sample S, compute from h some new positive examples $x_1, \ldots, x_q$ or more generally, some positive events $A_1, \ldots, A_q$, i.e. composed of positive examples; they are "characteristic" of h since they have a low Kolmogorov complexity relatively to h; draw some more examples from the target concept; if one of the x_i's is not drawn or if one of the A_i is not true, reject h.

We think that the previous heuristic is relevant from a cognitive point of view. Why from $a^2, a^8, a^{12}, a^6, \ldots$ don't we infer a^*? Doubtless because if the target was a^*, we would expect to see at least one odd exponent! That is, the reason why we reject the hypothesis $h = a^*$ is that the simple (characteristic) expected event "having an odd exponent" is not true.

In this paper, we study learning from simple positive examples and we show that learning algorithms can be based on this heuristic. Our main result is that *the class of regular languages is probably exactly learnable from simple positive examples*.

Our model is defined in Section 3; we define in Section 4 the notion of *positive teaching set* from which it is possible to reconstruct a given DFA; we show the existence of *simple* teaching sets in Section 5; eventually, we prove our main result in Section 6.

As a corollary, we show that the class of *simple regular languages* is learnable in the model of Li and Vitányi. This result generalizes the simple PAC learnability of simple k-reversible languages stated in Li and Vitányi (1991).

2. Preliminaries

2.1. Deterministic finite automata

Let Σ be a finite alphabet and Σ^* be the set of strings over Σ. For simplicity, we suppose here that $\Sigma = \{0, 1\}$ but all our results can be extended to the general case in a straightforward way. We write π_1 (resp. π_2) for the function which maps each element of $(\Sigma^*)^2$ on its first (resp. second) component (i.e., $\pi_1(x, y) = x$ and $\pi_2(x, y) = y$). We denote the null string by λ and the length of the string u by $|u|$. The size of a finite subset S of Σ^* is defined as $\|S\| = \sum_{u \in S} |u|$. We consider the ordering on Σ^* such that $u \prec v$ iff [$|u| < |v|$ or ($|u| = |v|$ and u is smaller than v in the lexicographical ordering)]. For every integer m, Σ^m (respectively $\Sigma^{\leq m}$) is the set of strings of length m (respectively of length at most m). A *language* is a subset of Σ^*. A language L is *prefix* if for all strings u and v, $u \in L$ and $uv \in L$ implies that $v = \lambda$. For every integer m, we let $L^m = L \cap \Sigma^m$ and $L^{\leq m} = L \cap \Sigma^{\leq m}$. For every string u, we let $u^{-1}L = \{v \in \Sigma^* \mid uv \in L\}$. If L_1 and L_2 are two languages, we let $L_1 \Delta L_2 = (L_1 \cup L_2) \backslash (L_1 \cap L_2)$.

A *deterministic finite automaton* (DFA) is a quintuple $A = (\Sigma, Q, q_0, T, \delta)$ where Q is a finite set of states, $q_0 \in Q$ is the initial state, $T \subseteq Q$ is the set of accepting states, and $\delta : Q \times \Sigma \to Q$ is the transition partial function. We still denote by δ the extended transition partial function defined over $Q \times \Sigma^*$. We denote the language accepted by A by $L(A)$, i.e. $L(A) = \{u \in \Sigma^* \mid \delta(q_0, u) \in T\}$. For every state $q \in Q$, we let $L_q = \{u \in \Sigma^* \mid \delta(q, u) \in T\}$. Thus, $L_{q_0} = L(A)$. A language is *regular* if it is recognized by a DFA. We write REG for the class of regular languages.

An automaton is said to be *trimmed* if every state is accessible from the initial state and if for every state q, there is a final state which is accessible from q. Given a DFA A, there is an efficient procedure to find a minimal DFA (in the number of states) for $L(A)$. We will say that a non empty regular language has n states to mean that the trimmed minimal DFA which recognizes it has n states.

Note that a DFA with n states can be encoded as a string of Σ^* of length $O(n \log n)$. An *encoding function* for DFA is a function $c : DFA \to \Sigma^*$ such that

- For every $A \in DFA$, $|c(A)|$ is bounded by $O(n \log n)$ and c runs in time polynomial in n, where $n = card(Q)$.
- For all automata $A, A' \in DFA$, A and A' are isomorphic iff $c(A) = c(A')$.

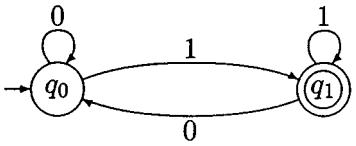
– There exists a polynomial-time deterministic algorithm which takes as input a string r and outputs an automaton A such that $c(A) = r$ if $r \in c(DFA)$ and *error* otherwise.
– For convenience, we will also suppose that the length of $c(A)$ is at least equal to the number of states of A, i.e. $|c(A)| \geq card(Q)$.

For example, if $A = (\Sigma, Q, q_0, T, \delta)$, we can take $c(A)$ to be the smallest string of the form

$$1^{card(Q)}0\bar{q}_0 1^{card(\delta)}0 \prod_{(q,x,q_1)\in\delta} \bar{q}x\bar{q}_1 1^{card(T)}0 \prod_{q\in T} \bar{q}$$

where $\bar{q}$ is a code for the state q whose length is $max(1, \lceil \log card(Q) \rceil)$.

Example 1. Let A be the minimal deterministic automaton which recognizes Σ^*1



With the previous scheme, 11001111000001110011110 1 is a code for A.

We fix such an encoding, say c.

2.2. *Kolmogorov complexity and Solomonoff-Levin distribution*

Complete definitions, results and proofs can be found in Li and Vitányi (1993).

Let $x, y \in \Sigma^*$. We let $\bar{x} = 1^{|x|}0x$ the *self-delimiting* version of x. We let $\langle x, y\rangle = \bar{y}x$. For $n \geq 3$, we let $\langle x_1, \ldots, x_n\rangle = \langle\langle x_1, \ldots, x_{n-1}\rangle, x_n\rangle$.

Let T be a Turing machine and $p, y \in \Sigma^*$. We write $T(\langle p, y\rangle) = x$ to mean that T with program p and extra information y terminates with output x. Let us define $K_T(x \mid y) = min\{|p| \text{ s.t. } T(\langle p, y\rangle) = x\}$, or ∞ if such p does not exist and $K_T(x) = K_T(x \mid \lambda)$. Therefore, $K_T(x \mid y)$ is the minimal length of a program that computes x from y.

We consider the prefix variant K of Kolmogorov complexity: a *prefix Turing machine* is a Turing machine for which the set of terminating programs is a prefix set. Note that for every string y and every prefix Turing machine T, the set of programs p such that $T(\langle p, y\rangle)$ halts is also a prefix set.

The Invariance Theorem states that there exists an *additively optimal* prefix Turing machine U, i.e. such that for any prefix Turing machine T, there is a constant c_T such that for all strings x, y, $K_U(x \mid y) \leq K_T(x \mid y) + c_T$. Therefore, for each pair of additively optimal prefix Turing machines U and U', there is a constant $c_{U,U'}$, such that for all strings x, y, $|K_U(x \mid y) - K_{U'}(x \mid y)| \leq c_{U,U'}$. Note that all the current programming languages are additively optimal prefix Turing machines.

We fix such a machine, say U, and call it the reference prefix Turing machine. We define $K(x) = K_U(x)$, and $K(x \mid y) = K_U(x \mid y)$.

It can be proved that for every strings x and r,

$$K(x \mid r) \leq |x| + 2\log|x| + O(1).$$

Each element x of a finite set of strings A can be identified by a program that computes A and its index in A. That is, if $Card(A) = n$,

$$K(x \mid r) \leq K(A \mid r) + \log n + 2\log\log n + O(1).$$

We will often use the following inequalities: for all strings x and y,

$$K(x) \leq K(\langle x, y\rangle) + O(1) \leq K(x) + K(y \mid x) + O(1) \leq K(x) + K(y) + O(1)$$

One of the most interesting property of the prefix Kolmogorov complexity is that for every string r,

$$\sum_{x \in \Sigma^*} 2^{-K(x \mid r)} < 1$$

Let $r \in \Sigma^*$, we define

$$\mathbf{m}_r(x) = \alpha_r 2^{-K(x \mid r)}$$

where $\alpha_r = (\sum_{x \in \Sigma^*} 2^{-K(x \mid r)})^{-1}$, and

$$\mathbf{m}(x) = \mathbf{m}_\lambda(x)$$

For each string r, $\mathbf{m}_r$ is a probability distribution on Σ^*.

The function $\mathbf{m}$ is called the Solomonoff-Levin distribution. It can be shown that $\mathbf{m}$ dominates every enumerable semi-distribution and that it is not computable.

2.3. Learning models

We will restrict here the classical definitions to the classes of languages.

Let $\mathcal{L}$ be a class of languages. A *representation scheme* R for $\mathcal{L}$ associates a set of names $R(L)$ with any language $L \in \mathcal{L}$. More formally, a representation scheme is a function $R : \mathcal{L} \to 2^{\Sigma^*}$ such that

- For every $L \in \mathcal{L}$, $R(L) \neq \emptyset$
- For all $L, L' \in \mathcal{L}$, if $L \neq L'$ then $R(L) \cap R(L') = \emptyset$
- There exists a polynomial-time deterministic algorithm which takes as input a pair of strings u and r and outputs 0 (resp. 1) if $r \in R(L)$ for some L and $u \notin L$ (resp. $u \in L$), and *error* otherwise.

We will use a *representation scheme* for the class of regular languages based on deterministic finite automata: for $L \in REG$, $R(L) = \{c(A) \mid A \in DFA \text{ and } L(A) = L\}$ where c is the encoding function for the DFA chosen above. For every non empty regular language L, we denote by r_L the encoding $c(A)$ of a minimal trimmed DFA A that recognizes L. Since we have supposed that isomorphic DFAs have the same code, r_L is correctly defined. As usual, the size of a regular language L is defined as $size(L) = min_{r \in R(L)}|r|$.

An *example* of a language L is a pair (u, e), where $u \in \Sigma^*$ and $e = 1$ if $u \in L$ and 0 otherwise. An example (u, e) is *positive* if $u \in L$ and *negative* otherwise. We denote by $EX(L)$ (respectively $POS(L)$) the set of all examples (respectively positive examples) of a language L. A *sample* (resp. a *positive sample*) of L is a finite subset of $EX(L)$ (resp. $POS(L)$). A *multisample* of L is a multiset of examples of L.

2.3.1. The learning model of Goldman and Mathias. The first learning model we consider is the model with teacher of Goldman and Mathias (1996). There are several reasons for considering it. First, regular languages are learnable in this model. Second, there are close connections between this model and the PACS model of learning that we mainly consider in this paper. Specifically, the learnability of REG in this model implies its learnability in PACS model. Third, the class of regular languages is not learnable from positive examples in this model while we prove it is in the PACS model.

Definition 1 (Goldman & Mathias, 1996). A class of language $\mathcal{L}$ is learnable in the model of Goldman and Mathias (GM learnable, in short) if there exist two algorithms *Teach* and *Learn* such that:

- Teach takes as input a (representation of a) language L and outputs a teaching sample $S_L \subseteq EX(L)$
- Learn takes as input any sample S such that $S_L \subseteq S \subseteq EX(L)$ and outputs a representation of L. In Goldman and Mathias (1996), the task of constituting S from S_L is entrusted to an Adversary whose goal is to prevent the learning.

If *Learn* runs in time polynomial in size(L) and $\|S\|$, the class $\mathcal{L}$ is *semi-poly GM learnable*. Moreover, if *Teach* runs in time polynomial in size(L), the class $\mathcal{L}$ *is polynomially GM learnable*.

It is proved in Goldman and Mathias (1996) that any class $\mathcal{L}$ learnable in deterministic polynomial time using example-based queries is semi-poly GM learnable. As it is proved in Angluin (1987) that REG is learnable from membership and equivalence queries, the class of regular languages is therefore semi-poly GM learnable. Moreover, it is not difficult to provide a polynomial teacher. A number of algorithms can be used to teach and learn DFAs in their model (Gold, 1978; Oncina & Garcia, 1992; Higuera, 1997).

For example, if we try to build an automaton incrementally from the initial state, we need to know whether a transition must arrive in a new state or come back in an already defined state. Therefore, a teaching algorithm can be based on this idea: for all states q and q' of the current automaton, and for every letter x, the teacher provides the learner with strings that

can differentiate $\delta(q, x)$ and q' if they must be. Then, the learning algorithm must merge any two states that cannot be distinguished.

Example 2. Let A be the minimal deterministic automaton which recognizes $L = \Sigma^*1$ (see Example 1).

We must differentiate q_0 from $q_1 = \delta(q_0, 1)$, q_0 from $\delta(q_1, 1)$ and q_1 from $\delta(q_1, 0)$.

Let $T(L) = \{(\lambda, -), (1, +), (11, +), (10, -)\}$. Obeying the order: "do not create a new state if it is not absolutely necessary", it is possible to reconstruct L incrementally from any set of examples that contains $T(L)$.

One problem with the learning model of Goldman and Mathias is that it is not sufficiently constrained to avoid collusion between the learner and the teacher. Nothing forbids the teacher to add the encoding of the target automaton (with the appropriate but useless label) to the teaching set. And nothing prevents the learner from using this information.

Let T be a polynomial teaching algorithm used to learn the class of regular languages in the GM model and let T' be a new "teacher" defined by:

$$T'(L) = T(L) \cup \{(r_L, e_L)\}$$

where r_L is the code for a minimal trimmed automaton that recognizes L and $e_L = 1$ if $r_L \in L$ and 0 otherwise.

Now consider the following algorithm

Input: a sample S which is a superset of the teaching set $T'(L)$
For every $u \in S$ **do**
  If u is the code of an automaton A **then**
    Let $L' = L(A)$
    Compute $T(L')$
    If L' is consistent with S **and if** $T(L') \subseteq S$ **then**
      halt
Output: a representation of L'

Since there exists a learning algorithm that computes any regular language L from any sample S containing $T(L)$, there only can be one regular language L consistent with a sample S that contains $T(L)$. Hence, L' is equal to L. Therefore, every learning algorithm for REG can be rendered trivial. This does not mean that all learning algorithms in the GM model are collusive but that, if there exist polynomial teaching and learning algorithms and if target concepts can be encoded by examples, then there must also exist collusive learning algorithms.

2.3.2. PAC learning. We assume familiarity with the basic facts about PAC learning theory, see for example Kearns and Vazirani (1994), Natarajan (1991a), and Valiant (1984).

Let $\mathcal{L}$ be a class of language, let $L \in \mathcal{L}$ be a target language and let μ be any fixed probability distribution over Σ^*. Let $EX(L, \mu)$ be a procedure that runs in unit time and that at each call returns an example (u, e) of L, where u is drawn randomly and independently

according to μ. If L' is any concept in $\mathcal{L}$, we define *error*(L') (with respect to L and μ) to be the probability that L' is inconsistent with an example of L drawn randomly according to μ (i.e. *error*$(L') = \mu(L \Delta L')$).

Definition 2. Let $\mathcal{L}$ be a class of language and let R be a representation scheme for $\mathcal{L}$.

- An algorithm $\mathcal{A}$ is a PAC *learning algorithm* for $\mathcal{L}$ in R if $\mathcal{A}$ takes as input $\epsilon \in (0, 1]$, $\delta \in (0, 1]$, an integer l, and for any language L in $\mathcal{L}$ with size$(L) \leq l$ and any probability distribution μ, $\mathcal{A}$ is given access to $EX(L, \mu)$ and $\mathcal{A}$ outputs the representation of some L' in $\mathcal{L}$, such that with probability at least $1 - \delta$, error$(L') \leq \epsilon$.
- $\mathcal{L}$ is *PAC learnable* if there is a PAC learning algorithm $\mathcal{A}$ for $\mathcal{L}$ which runs in time polynomial in $1/\epsilon$, $1/\delta$, l, and the size of the longest example that has been drawn.

The parameter l can be omitted from the input parameters and guessed by the learning algorithm since the learner can test whether a hypothesis is good enough for the learning task (Haussler et al., 1991).

Previous definition can be relaxed in the following way: the output of the learning algorithm can belong to a larger hypothesis class. We say that a class of language is *predictable* if it is PAC learnable in *some* polynomially evaluatable hypothesis class.

The main result concerning the PAC learnability of the class of regular language is negative: assuming some cryptographic assumptions, REG is not predictable.

Theorem 1 (Kearns & Valiant, 1994; Kearns & Vazirani, 1994)**.** *Under the discrete root assumption, the representation class of DFA is inherently unpredictable.*

This result is not isolated. It has turned out that many learning problems are intractable under standard PAC model. What is the reason of this fact? In the PAC learning model, the algorithm must learn under *all* distributions. This is called the *distribution free requirement*. But can we hope to learn a language if it shares no relation with the distribution that provides examples? It has been proved that limiting the allowed distributions could improve drastically the expressivity of the model. But how to limit the allowed distribution in a not too restrictive and unnatural way? The simple PAC learning models we present in the next section make the hypothesis that examples with low Kolmogorov complexity have a higher weight in the learning process.

2.3.3. Simple PAC learning models. A variant of the PAC learning model in which the class of probability distributions is restricted to the universal Solomonoff-Levin distribution **m** has been defined in Li and Vitányi (1991).

> "The remarkable property of this distribution is that in a polynomial sample with overwhelming probability all examples of logarithmic complexity (the simple examples) will be represented. Hence, a learning algorithm simply needs to reconstruct a concept approximatively when all simple examples are given." (Li & Vitányi, 1993)

In Denis et al. (1996), we have defined another simple PAC learning model. See Denis and Gilleron (1997a) for a study of the connections between these two models. The underlying idea is to suppose that the oracle knows a representation of the target concept and gives a higher weight to the examples which are *simple according to this representation*.

The definition we give here is slightly more general than in Denis et al. (1996). We define for each language L a set of admissible distribution D_L and we suppose that the distribution used to draw the examples is admissible.

Definition 3. Let $\mathcal{L}$ be a class of languages and let R be a representation scheme for $\mathcal{L}$. For every language $L \in \mathcal{L}$, we define the set of admissible distributions according to R as

$$D_L = \{\mathbf{m}_r \mid r \in R(L)\}$$

Loosely speaking, an example x of L is simple according to the representation $r \in R(L)$ if $K(x \mid r) = O(\log size(L))$. A polynomial sample will contain all simple examples with probability almost 1. Moreover, if each language has a canonical representation r_L, i.e. if there exists an algorithm that computes r_L from any representation $r \in R(L)$ for any language L, a polynomial sample drawn according to any admissible distribution $\mathbf{m}_r$ will almost certainly contain all the simple examples according to r_L. Indeed, $K(r_L \mid r) = O(1)$ since there exists a program that computes r_L from r and every simple example according to r_L is also simple according to r since

$$K(x \mid r) \leq K(x \mid r_L) + K(r_L \mid r) + O(1) \leq K(x \mid r_L) + O(1)$$

Definition 4. Let $\mathcal{L}$ be a class of languages and let R be a representation scheme for $\mathcal{L}$.

- An algorithm $\mathcal{A}$ is a *simple PAC learning algorithm for $\mathcal{L}$ in* R if $\mathcal{A}$ takes as input $\epsilon \in (0, 1]$, $\delta \in (0, 1]$, an integer l, and for any language L in $\mathcal{L}$ with size$(L) \leq l$ and any admissible distribution μ, $\mathcal{A}$ is given access to $EX(L, \mu)$ and $\mathcal{A}$ outputs the representation of some L' in $\mathcal{L}$, such that with probability at least $1 - \delta$, error$(L') \leq \epsilon$.
- $\mathcal{L}$ is *PAC learnable from simple examples* in R (PACS learnable, in short) if there is a simple PAC learning algorithm $\mathcal{A}$ for $\mathcal{L}$ in R which runs in time polynomial in $1/\epsilon$, $1/\delta$ and l.

Note that since there are arbitrarily long simple examples (such as $0^{2^{2^{|r|}}}$), the size of the longest example seen cannot appear among the parameters kept for measuring the time complexity of a learning algorithm. Otherwise, retaining the examples seen would always be a sufficient learning strategy! But note also that the hypothesis testing procedure designed in Haussler et al. (1991) cannot be used anymore since too long examples cannot be handled. Therefore, it seems impossible to omit the input parameter l.

The Kolmogorov complexity of a string depends on the reference Turing machine that has been chosen. It is not known whether the set of PACS learnable classes depends too on the reference Turing machine. But the independence can be easily verified for all the classes which have been proved learnable in this model: it is sufficient to verify that no particular property of the reference machine is used.

In the general case, it is not conceivable to require a standard PAC algorithm to output exactly the target language. Indeed, if most of the examples have a zero weight under the underlying distribution, there is no way to identify exactly the target. But as the Solomonoff-Levin distribution puts a positive weight on any example, we will say that a learning algorithm is *probably exactly correct* if it outputs a representation of the target with high confidence.

Definition 5. Let $\mathcal{L}$ be a class of languages and let R be a representation scheme for $\mathcal{L}$.

- An algorithm $\mathcal{A}$ is a *simple PEC learning algorithm for* $\mathcal{L}$ *in* R if $\mathcal{A}$ takes as input $\delta \in (0, 1]$, an integer l, and for any language L in $\mathcal{L}$ with size$(L) \leq l$ and any admissible distribution μ, $\mathcal{A}$ is given access to $EX(L, \mu)$ and $\mathcal{A}$ outputs the representation of some L' in $\mathcal{L}$, such that with probability at least $1 - \delta$, $L = L'$.
- $\mathcal{L}$ is *PEC learnable from simple examples* in R if there is a simple PEC learning algorithm $\mathcal{A}$ for $\mathcal{L}$ in R which runs in time polynomial in $1/\delta$ and l.

Connections between GM learnability and PEC learnability have been studied in Denis and Gilleron (1997b). Castro and Guijarro have studied connections between this model and exact learning with queries (Castro & Guijarro, 1998). DFA have been proved PACS learnable in Parekh and Honavar (1997).

Proposition 1. *If a class of language $\mathcal{L}$ is polynomially GM learnable, then it is PEC learnable from simple examples.*

Proof: Sketch. Let T be a teaching algorithm. Since T runs in polynomial time, there exists a integer k such that for every language $L \in \mathcal{L}$, $Card(T(L)) \leq (size(L))^k$. Let r be a representation of L. Since $T(L)$ is computable from r, any element in $T(L)$ can be computed from r by a program whose length is $O(\log size(L))$. Hence, for all $x \in T(L)$, we have $K(x \mid r) \leq O(\log size(L))$, i.e. every example in $T(L)$ is simple, according to a representation of the target language. Therefore, a polynomial sample drawn with an admissible distribution will contain the whole teaching set $T(L)$ with high probability. □

Corollary 1. *The class of DFA is PEC learnable from simple examples.*

The PACS model and the GM model share the same defect: if it is possible to encode directly the target by means of examples, learning can be rendered trivial. That is, the model is not strong enough to avoid collusion between learner and teacher. We must verify in each case that such a cheating cannot be employed!

2.4. Learning from positive examples

The importance of learning from positive data in natural learning is well recognized. Namely, as it is reported in Gold seminal paper (Gold, 1967), the acquisition of one's mother tongue is based on sentences which are syntactically correct. Against this assertion, it is sometimes

argued that an incorrect sentence uttered by a child which has no effect in return could play the role of a negative example. But this only means that the learner is able to *simulate* some negative data in comparing provided and expected information.

What is it possible to do in grammatical inference on the basis of positive examples? Gold proved in 1967 that a class of languages that contains all of the finite languages and at least one infinite language cannot be identified in the limit from positive examples. As a corollary, regular languages are not learnable in Gold's model. As natural languages are certainly more complex than regular languages, the non learnability of REG raises a problem. Gold suggests several ways to solve this contradiction. We will consider one of them in this paper: examples are not provided to the learner in an arbitrary way.

Why is learning from positive data so difficult? The main reason is that a too general hypothesis will never be refuted from a new positive example. A learning algorithm has classically to prevent *overspecialization*. A learning algorithm which has to learn from positive data has to prevent both overspecialization and *overgeneralization*.

Note the result of Angluin (1980) that *characterizes* the indexed families of recursive languages which are identifiable in the limit from positive examples. Loosely speaking, we can give the following interpretation of her result: in order to learn from positive data, it is necessary to associate a (computable or enumerable) characteristic sample S_L to each hypothesis L. A current hypothesis L, consistent with the current positive sample S will nevertheless be refuted if $S_L \not\subseteq S$. This provides a general way to prevent overgeneralization. The heuristic we will use in our learning algorithm is based on this general principle.

As every GM learnable class is identifiable in the limit, the class REG is not learnable from positive examples in the learning model of Goldman and Mathias. There is a very easy direct proof: suppose that REG is GM learnable from positive examples. Let *Teach* and *Learn* be the teaching and learning algorithms. Let L be an infinite regular language and let $L' = \mathit{Teach}(L)$. The language L' is regular itself. We have $\mathit{Teach}(L') \subseteq L' = \mathit{Teach}(L) \subset L$. We must have $\mathit{Learn}(\mathit{Teach}(L)) = L$ and $\mathit{Learn}(\mathit{Teach}(L)) = L'$, which is contradictory.

Unfortunately, the situation is even getting worse in the PAC learning model. It can be easily shown that if a class is PAC learnable from positive data (as the class of k-CNF), the output hypothesis must be included in the target concept. But as it is impossible from positive data to differentiate a negative example from an absent positive one, even very simple classes cannot be PAC learnable from positive data. See Natarajan (1991b), Shvayster (1990), and Denis (1998) for a detailed study.

See also Sakakibara (1992) for results on grammatical inference about learning from *structured positive examples* (Kanazawa, 1996) for *identification in the limit of categorial grammars* and (Tellier, 1998) for *syntactico-semantic searning of categorical grammars*.

3. Learning from simple positive examples

There are *a priori* reasons to think that even complex classes can be learned from positive *simple* examples.

Suppose that from some correct, simple, frequent sentences such as

"Humphrey runs."
"Humphrey meets Lauren"
"He runs."
"He meets Lauren"

a learner infers that the words "Humphrey" and "he" belong to the same syntactical category.

If correct sentences such as

"Lauren meets Humphrey"
"Lauren talks to Humphrey"

are sufficiently frequent, the absence of sentences of the form

"*Lauren meets he"
"*Lauren talks to he"

might be sufficient to make the learner give up his hypothesis. They are so simple that if they were correct, they would have be produced.

In the PACS learning model, a characteristic example of the target is an example which can be computed by a small program, the target being known. If the learner can efficiently compute characteristic examples of his current hypothesis and if these examples do not appear within reasonable time in the current sample, he has good reasons to discard his hypothesis, since a characteristic example must be frequent too. In other words, the fact that simple examples have a higher weight under the underlying distribution might provide a heuristic to prevent overgeneralization. The goal of this paper is to show that this heuristic can really be used to learn regular languages.

First, we must adjust the Definition 4.

Definition 6. Let $\mathcal{L}$ be a class of languages and let R be a representation scheme for $\mathcal{L}$. For every non empty language $L \in \mathcal{L}$, we define the set of *positive admissible distributions* according to R as

$$D_L^+ = \{\mu_r \mid r \in R(L)\}$$

where

$$\mu_r(x) = \begin{cases} 0 & \text{if } x \notin L, \\ \dfrac{\mathbf{m}_r(x)}{\mathbf{m}_r(L)} & \text{if } x \in L \end{cases}$$

where $\mathbf{m}_r(L) = \sum_{x \in L} \mathbf{m}_r(x)$.

That is, μ_r is the positive restriction of an admissible distribution for the target language L. Note that for every $x \in L$, $\mu_r(x) \geq \mathbf{m}_r(x)$.

Definition 7. Let $\mathcal{L}$ be a class of languages and let R be a representation scheme for $\mathcal{L}$.

– An algorithm $\mathcal{A}$ is a *probably exactly correct* (*PEC*) *learning algorithm from simple positive examples for* $\mathcal{L}$ *in* R if $\mathcal{A}$ takes as input $\delta \in (0, 1]$, an integer l, and for any non empty language L in $\mathcal{L}$ with size$(L) \leq l$ and any positive admissible distribution μ, $\mathcal{A}$ is given access to $EX(L, \mu)$ and $\mathcal{A}$ outputs the representation of some L' in $\mathcal{L}$, such that with probability at least $1 - \delta$, $L = L'$.
– $\mathcal{L}$ is *PEC learnable from simple positive examples* in R if there is a PEC learning algorithm from simple positive examples $\mathcal{A}$ for $\mathcal{L}$ in R which runs in time polynomial in $1/\delta$ and l.

Roughly speaking, a *simple positive example of a language* L is a positive example of low Kolmogorov complexity according to the representation r_L of the minimal trimmed automaton which recognizes L. Before to get to the heart of the matter, we show below that a polynomial sized (sufficiently large) sample drawn according to *any* positive admissible distribution will almost certainly contain the simple positive examples of L. The proof is based on several technical lemmas.

Lemma 1. *Recall that for any strings* r *and* x, $\mathbf{m}_r(x)$ *is defined as* $\alpha_r 2^{-K(x\,|\,r)}$. *There exists a constant* c_0 *such that for every string* r,

$$1 < \alpha_r \leq c_0$$

Proof: Since for every string r, $\sum_{x \in \Sigma^*} 2^{-K(x\,|\,r)} < 1$, we get $1 < \alpha_r$.
There exists a constant β such that for every string r, $K(r\,|\,r) \leq \beta$. Then,

$$1 \geq \mathbf{m}_r(r) = \alpha_r 2^{-K(r\,|\,r)} \geq \alpha_r 2^{-\beta}$$

i.e. $\alpha_r \leq 2^{\beta}$. Take $c_0 = 2^{\beta}$. □

The following lemma shows that if L is a non empty regular language and if r is a representation for L then $\mathbf{m}_r(L)$ has a non null lower bound.

Lemma 2. *Let* $\mathcal{L}$ *be a class of languages and let* R *be a representation scheme for* $\mathcal{L}$. *There exists a constant* c_1 *such that for every non empty language* L *in* $\mathcal{L}$, *and every representation* $r \in R(L)$, *we have*

$$\mathbf{m}_r(L) \geq c_1$$

So, for every $x \in L$, *we get*

$$2^{-K(x\,|\,r)} < \mathbf{m}_r(x) \leq \mu_r(x) \leq c_1^{-1}\mathbf{m}_r(x) \leq c_0 c_1^{-1} 2^{-K(x\,|\,r)}$$

Proof: Let us write *first*(L) for the first string in L (in the $\prec$ ordering). There exists a program which takes as input any representation r of any non empty language $L \in \mathcal{L}$, and outputs *first*(L). Then,

$$\exists \beta\ \forall L \in \mathcal{L}, \forall r \in R(L), K(\textit{first}(L)\,|\,r) \leq \beta$$

Therefore,

$$\mathbf{m}_r(L) \geq \mathbf{m}_r(\mathit{first}(L)) > 2^{-K(\mathit{first}(L)\,|\,r)} \geq 2^{-\beta}$$

Let $c_1 = 2^{-\beta}$: we get the result. □

Constants c_0 and c_1 will be used throughout this paper.

We show in the two next lemmas that if r_L is the encoding of a minimal DFA that recognizes L, the distributions $\mathbf{m}_{r_L}$ and μ_{r_L} are kinds of minimal elements (up to a multiplicative constant) in the sets of admissible (respectively positive admissible) distributions for L.

Lemma 3. *There exists a constant c such that for every non empty regular language L, for every representation $r \in R(L)$ and for every string x,*

$$\mathbf{m}_r(x) \geq c\mathbf{m}_{r_L}(x) \text{ and } \mu_r(x) \geq c\mu_{r_L}(x)$$

Proof: There exists a program which takes as input the representation r of a regular language L, computes the corresponding automaton A, minimizes it and outputs the representation r_L in R of the equivalent minimal DFA.

That is, there exists a constant γ such that for every non empty regular language L, for every representation $r \in R(L)$ and for every string x, $K(x\,|\,r) \leq K(x\,|\,r_L) + \gamma$.

We have

$$\mathbf{m}_r(x) > 2^{-K(x\,|\,r)} \geq 2^{-K(x\,|\,r_L)-\gamma}$$

and then

$$\mathbf{m}_r(x) \geq 2^{-\gamma}\mathbf{m}_{r_L}(x)\left(\alpha_{r_L}\right)^{-1} \geq 2^{-\gamma}\mathbf{m}_{r_L}(x)c_0^{-1}$$

From previous lemma, we get

$$\mu_r(x) \geq \mathbf{m}_r(x) \geq 2^{-\gamma}\mathbf{m}_{r_L}(x)c_0^{-1} \geq 2^{-\gamma}c_0^{-1}c_1\mu_{r_L}(x)$$

Let $c = 2^{-\gamma}c_0^{-1}c_1$: we get the result. □

Proposition 2. *For all integers k, l, for every $\delta > 0$, for any non empty regular language L such that $\mathit{size}(L) \leq l$ and for every $r \in R(L)$, it is sufficient to draw $O(kl^k \ln(l/\delta))$ examples according to μ_r to get all the elements x of L such that $K(x\,|\,r_L) \leq k\log(\mathit{size}(L))$ with a confidence greater that $1-\delta$.*

Proof: Since the number of programs of size $\leq h$ is less than 2^{h+1}, the number of elements x such that $K(x\,|\,r_L) \leq k\log(\mathit{size}(L))$ is less than $2(\mathit{size}(L))^k \leq 2l^k$.

Applying previous results, it is easy to show that there exists a constant c such that for all x in L, we have

$$\mu_r(x) \geq c2^{-K(x\,|\,r_L)}$$

i.e. if $K(x \mid r_L) \leq k \log(size(L)) \leq k \log l$

$$\mu_r(x) \geq cl^{-k}$$

The probability that one such element of L is not drawn after N independent draws is less than

$$(1 - cl^{-k})^N$$

The probability that there exists one such element of L that is not drawn after N independent draws is less than

$$2l^k(1 - cl^{-k})^N$$

As for $x \leq 1$ we have $(1 - x)^N \leq e^{-Nx}$, verify that if

$$N \geq \frac{l^k}{c} \ln \frac{2l^k}{\delta}$$

we have

$$2l^k(1 - cl^{-k})^N \leq \delta$$ □

4. Learning regular languages from positive teaching sets

We define below the notion of *positive teaching set* for a language L. These sets are designed to satisfy an essential property: a non empty regular language L is polynomially identifiable from a positive teaching set for L.

Definition 8. Let L be a non empty regular language and let $A = (\Sigma, Q, q_0, T, \delta)$ be a trimmed minimal DFA which recognizes L. A finite set $S \subset (\Sigma^*)^2$ *is a positive teaching set* for L if

1. $S = \{(u, v) \in \pi_1(S) \times \pi_2(S) \mid uv \in L\}$,
2. $\lambda \in \pi_1(S)$ and $\lambda \in \pi_2(S)$,
3. For every state $q \in Q$, there exists $u \in \pi_1(S)$ such that $\delta(q_0, u) = q$,
4. For every pair of distinct states $q, q' \in Q$, there exists $v \in \pi_2(S) \cap (L_q \Delta L_{q'})$,
5. For every state $q \in Q$ and every letter $x \in \Sigma$, if $q' = \delta(q, x)$ is defined then there exists $u \in \pi_1(S)$ such that $\delta(q_0, u) = q$ and $ux \in \pi_1(S)$,

Note that this notion is not very robust since a superset of a positive teaching set for L is not necessarily a positive teaching set. However, we have:

Lemma 4. *If S' is a superset of a positive teaching set S for L and if S' satisfies item 1 above, then S' is a positive teaching set for L.*

Proof: Straightforward since $\pi_1(S) \subseteq \pi_1(S')$ and $\pi_2(S) \subseteq \pi_2(S')$ □

Example 3. Let A be the trimmed minimal DFA which recognizes Σ^*1 (see Example 1).

The set $S = \{(\lambda,1), (\lambda,01), (0,1), (0,01), (1,\lambda), (1,1), (1,01), (10,1), (10,01), (11,\lambda), (11,1), (11,01)\}$ is a positive teaching set for L.

We show below how to build a particular positive teaching set. But first, we need the following lemma:

Lemma 5. *Let L be a non empty regular language and let $A = (\Sigma, Q, q_0, T, \delta)$ be the trimmed minimal DFA which recognizes L. For every non empty $R \subseteq Q$, there exists $V_R \subset \Sigma^*$ and a one-to-one mapping $\sigma_R : V_R \to R$ such that*

- *$card(V_R) < card(R)$,*
- *For every pair of distinct states $q, q' \in R$, there exists $v \in V_R$ such that $v \in L_q \Delta L_{q'}$,*
- *For all $v \in V_R$, we have $\delta(\sigma_R(v), v) \in T$.*

Proof: We prove this lemma by induction on $card(R)$. The result is clear for $card(R) = 1$. Suppose that $card(R) \geq 2$.

Let v_0 be the first string in Σ^* such that there exist two distinct states $q, q' \in R$ such that $v_0 \in L_q \Delta L_{q'}$.

Let $R' = \{q \in R \mid v_0 \in L_q\}$ and let $R'' = \{q \in R \mid v_0 \notin L_q\}$. As R' and R'' are strictly included in R, we can apply the induction hypothesis. Let $V_{R'}$, $V_{R''}$, $\sigma_{R'}$ and $\sigma_{R''}$ be the corresponding sets and mappings.

Define $V_R = V_{R'} \cup V_{R''} \cup \{v_0\}$. We have $card(V_R) \leq card(V_{R'}) + card(V_{R''}) + 1 < card(R)$. Verify that for any distinct states $q, q' \in R$, there exists $v \in V_R$ such that $v \in L_q \Delta L_{q'}$.

Now, let q' be the first state in $R' \backslash \sigma_{R'}(V_{R'})$ (i.e. such that $\exists u \delta(q_0, u) = q'$ and $\forall u' \neq u, \delta(q_0, u') \in R' \backslash \sigma_{R'}(V_{R'}) \Rightarrow u \prec u'$). For every $v \in V_R$, define

$$\sigma_R(v) = \begin{cases} \sigma_{R'}(v) & \text{if } v \in V_{R'}, \\ \sigma_{R''}(v) & \text{if } v \in V_{R''} \backslash V_{R'}, \\ q' & \text{if } v = v_0 \text{ and } v \notin V_{R'} \cup V_{R''} \end{cases}$$

σ_R is clearly one-to-one and $\forall v \in V_R, \delta(\sigma_R(v), v) \in T$. □

Definition 9. Let L be a non empty regular language, let $A = (\Sigma, Q, q_0, T, \delta)$ be the trimmed minimal DFA which recognizes L and let $n = card(Q)$. For every state $q \in Q$, let us write u_q for the first string (in the $\prec$ ordering) such that $\delta(q_0, u_q) = q$. Let $U_A = \{u_q \mid q \in Q\} \cup \{u_q x \mid q \in Q, x \in \Sigma \text{ and } \delta(q, x) \text{ is defined}\}$. Now, let V_Q and σ_Q as defined in the previous lemma. For every $q \in Q$, let

$$v_q = \begin{cases} v & \text{if } \sigma_Q(v) = q \\ \text{the first string in } L_q & \text{otherwise} \end{cases}$$

Now let $V_A = \{v_q \mid q \in Q\}$ and $S_A = \{(u, v) \in U_A \times V_A \mid uv \in L\}$.

Lemma 6.
1. *U_A has at most $2n+1$ elements, $u_{q_0} = \lambda$ and the length of every element in U_A is at most n;*
2. *V_A has at most n elements, $\lambda \in V_A$ and the length of every element in V_A is at most $n-1$.*

Proof:

1. Associate $u_q x$ to each transition $(q, x, q') \in \delta$; if $u_q x = u_{q_1} x_1$ then $x = x_1$, $q = q_1$ and $\delta(q, x) = \delta(q_1, x_1)$; as each element in U_A except λ corresponds to one transition, $card(U_A) \leq 2n+1$. The two other points are straightforward.
2. It is clear that $card(V_A) \leq n$; as the length of the least string that distinguishes two states is at most $n-1$, the length of each element in V_A is at most $n-1$. Now, let us prove that $\lambda \in V_A$: if all states in Q are terminal states, for every $q \in Q \backslash V_Q$, $v_q = \lambda$; and if not all states are terminal, the first element in V_Q is λ.

□

As a corollary, we get:

Proposition 3. *Let L be a non empty regular language and let $A = (\Sigma, Q, q_0, T, \delta)$ be the trimmed minimal DFA which recognizes L. S_A is a positive teaching set for L that we will designate as the canonical positive teaching set for L.*

Example 4. Let A be the trimmed minimal DFA which recognizes $\Sigma^* 1$. We have $U_A = \{\lambda, 0, 1, 10, 11\}$, $V_Q = \{\lambda\}$, $V_A = \{\lambda, 1\}$, $S_A = \{(\lambda, 1), (0, 1), (1, \lambda), (1, 1), (10, 1), (11, \lambda), (11, 1)\}$.

Proposition 4. *There exists a polynomial algorithm $\mathcal{A}$ such that for any non empty regular language L and any positive teaching set S for L, if $\mathcal{A}$ takes S as input then $\mathcal{A}$ outputs a trimmed minimal DFA for L.*

Proof: Let L be a non empty regular language, let $A = (\Sigma, Q, q_0, T, \delta)$ be a minimal trimmed automaton which recognizes L and let S be a positive teaching set for L.

Let $\sim_S$ be the relation defined over $\pi_1(S)$ by

$$u \sim_S v \text{ iff } \forall\, w \in \Sigma^* \ (u, w) \in S \Leftrightarrow (v, w) \in S$$

1. The relation $\sim_S$ is an equivalence relation. Let us write $\bar{u}$ the class of the string u and let Q' be the quotient set of $\pi_1(S)$ modulo to $\sim_S$.
2. For all u and v in $\pi_1(S)$, $\delta(q_0, u) = \delta(q_0, v) \Leftrightarrow \bar{u} = \bar{v}$.
3. Let $\phi : Q' \to Q$ defined by $\phi(\bar{u}) = q$ such that $\delta(q_0, u) = q$. Previous item ensures that this definition is correct. Verify that
   - ϕ is one-to-one onto Q,
   - $\phi(\bar{\lambda}) = q_0$,
   - $\phi(\bar{u}) \in T$ iff $(u, \lambda) \in S$,

– If $\delta(q, x)$ is defined, there exists $u \in \pi_1(S)$ such that $ux \in \pi_1(S)$, $\phi(\overline{u}) = q$ and $\phi(\overline{ux}) = \delta(q, x)$.

Now, it is easy to check that the following algorithm is well defined, that it runs in polynomial time and that the output automaton is a DFA isomorphic to A.

Algorithm $\mathcal{A}$
Input: S, a positive teaching set for some regular language L
Defining the set of states Q'
$Q' \leftarrow \emptyset$
For every $u \in \pi_1(S)$ **do**
if there exist $q \in Q'$ **and** $u' \in q$ such that $u \sim_S u'$
then
$q \leftarrow q \cup \{u\}$
else
$Q' \leftarrow Q' \cup \{\{u\}\}$
Defining the initial state q'_0
there exists a unique element $q'_0 \in Q'$ such that $\lambda \in q'_0$
Defining the set of final states T'
$T' \leftarrow \{q \in Q' \mid \exists u \in q \text{ and } (u, \lambda) \in S\}$
Defining the transition function δ'
For all $q \in Q'$ **and for all** $x \in \Sigma$
if there exist u **and** $q' \in Q'$ such that $u \in q$ and $ux \in q'$
then
$\delta'(q, x) = q'$
Let $A' = (\Sigma, Q', q'_0, T', \delta')$ and $r = c(A')$
Output: r □

It is not very surprising that we could rebuild a regular language from a positive teaching set: this notion has been designed for this purpose. We show in the next section that there are natural ways to find such sets.

5. Simple positive teaching sets

It is not too difficult to build a positive teaching set whose elements have simple components. It is more interesting to remark that a set of the form $\{(u, v) \mid uv \in L\}$ is a positive teaching set as soon as its elements have (sufficiently) simple components.

Lemma 7. *There exists a constant c_2 such that for every non empty regular language L with n states, for every $r \in R(L)$, for every pair $(u, v) \in S_A$ (where S_A is the canonical positive teaching set for L), we have*

$$K(u \mid r) \leq \log n + 2 \log \log n + c_2 \text{ and } K(v \mid r) \leq \log n + 2 \log \log n + c_2$$

We will use such a constant c_2 throughout this paper.

Proof: There exists a program that builds from r the corresponding trimmed minimal DFA A which recognizes L. Therefore, there exists a program that computes U_A from r (see Definition 9). Each element of U_A can be be identified from r by its index in U_A. As U_A has at most $2n+1$ elements, each of its element can be identified from r by a program whose length is at most $\log n + 2\log\log n + O(1)$ (see Section 2.2).

The second inequality is proved in a similar way. □

Lemma 8. *There exists a constant c_3 such that for every non empty regular language L, for every $r \in R(L)$ and for every string $u \in \Sigma^*$ such that $K(u \mid r) \leq \log n + 2\log\log n + c_2$, we have*

$$\text{if } u\Sigma^* \cap L^{\leq 2n-1} \neq \emptyset \text{ then } \mu_r(u\Sigma^* \cap L^{\leq 2n-1}) \geq \frac{c_3}{n(\log n)^2}$$

and

$$\text{if } \Sigma^* u \cap L^{\leq 2n-1} \neq \emptyset \text{ then } \mu_r(\Sigma^* u \cap L^{\leq 2n-1}) \geq \frac{c_3}{n(\log n)^2}$$

We will use such a constant c_3 throughout this paper.

Proof: Suppose that $u\Sigma^* \cap L^{\leq 2n-1} \neq \emptyset$. There exists a program that, with inputs r and u finds the first string w such that $uw \in L^{\leq 2n-1}$. Then, there exists a constant γ_1 such that

$$K(uw \mid r) \leq K(u \mid r) + \gamma_1$$

We have

$$\mu_r(u\Sigma^* \cap L^{\leq 2n-1}) \geq \mu_r(uw) \geq 2^{-K(uw \mid r)} \geq 2^{-K(u \mid r)-\gamma_1} \geq \frac{2^{-\gamma_1 - c_2}}{n(\log n)^2}$$

In a similar way we prove that there exists a constant γ_2 such that

$$\text{if } \Sigma^* u \cap L^{\leq 2n-1} \neq \emptyset \text{ then } \mu_r(\Sigma^* u \cap L^{\leq 2n-1}) \geq \frac{2^{-\gamma_2 - c_2}}{n(\log n)^2}$$

Let $c_3 = \min(2^{-\gamma_1 - c_2}, 2^{-\gamma_2 - c_2})$; we get the result. □

Definition 10. Let L be a non empty regular language with n states. Let $a > 0$ and let $r \in R(L)$. We let

$$\begin{aligned} U_r^a &= \left\{ u \in \Sigma^{\leq n}, \mu_r(u\Sigma^* \cap L^{\leq 2n-1}) \geq a \right\} \\ V_r^a &= \{ v \in \Sigma^{\leq n-1}, \mu_r(\Sigma^* v \cap L^{\leq 2n-1}) \geq a \} \\ S_r^a &= \left\{ (u, v) \in U_r^a \times V_r^a \;\middle|\; uv \in L \right\} \end{aligned}$$

Lemma 9. *We have*

$$\begin{aligned} Card\big(U_r^a\big) &\leq \frac{n+1}{a} \\ Card\big(V_r^a\big) &\leq \frac{n}{a} \\ Card\big(S_r^a\big) &\leq \frac{n(n+1)}{a^2} \end{aligned}$$

Proof: Let us write $fr(U_r^a)$ for the set of strings of U_r^a which are not prefix of another string in U_r^a. That is

$$fr\big(U_r^a\big) = \big\{u \in U_r^a \;\big|\; \forall u' \in U_r^a, u' \in u\Sigma^* \Rightarrow u = u'\big\}$$

$fr(U_r^a)$ is a prefix set. Therefore, if $u, u' \in fr(U_r^a)$, $u \neq u' \Rightarrow u\Sigma^* \cap u'\Sigma^* = \emptyset$. We have:

$$\begin{aligned} 1 &\geq \mu_r\big(\cup\big\{u\Sigma^* \mid u \in fr\big(U_r^a\big)\big\}\big) = \Sigma\big\{\mu_r(u\Sigma^*) \;\big|\; u \in fr\,\big(U_r^a\big)\big\} \\ &\geq a \times Card\big(fr\big(U_r^a\big)\big) \end{aligned}$$

Therefore $Card(fr(U_r^a)) \leq 1/a$
In the other hand, every string of length n has at most $n+1$ prefixes. Then

$$Card\big(U_r^a\big) \leq \frac{n+1}{a}$$

We prove in a similar way that $Card(V_r^a) \leq n/a$.
Lastly,

$$Card\big(S_r^a\big) \leq Card\big(U_r^a\big) \times Card\big(V_r^a\big) \leq \frac{n(n+1)}{a^2}$$

□

Proposition 5. *Let L be a non empty regular language with n states. Let $r \in R(L)$ and let a be such that $0 < a \leq c_3/n(\log n)^2$ (where c_3 is defined in Lemma 8). Then, S_r^a is a positive teaching set and its size is polynomial in n and $1/a$.*

Proof: Applying Lemmas 7 and 8, we see that S_r^a is a superset of S_A. From Lemma 4, S_r^a is a positive teaching set. From Lemma 9, the cardinal of S_r^a is polynomial in n and $1/a$.

□

Corollary 2. *Let L be a non empty regular language with n states, and let a be such that $0 < a \leq c_3/n(\log n)^2$. We can compute from S_r^a a minimal DFA which recognizes L in time polynomial in n and $1/a$.*

Proof: Apply Propositions 4 and 5. □

We have proved that every regular language has simple positive teaching sets. It remains to show that it is possible to find such a teaching set from simple positive examples.

6. Learning regular languages from simple positive examples

Lemma 8 shows that if a word $uv \in L$ has simple components u and v, then the events $u\Sigma^* \cap L^{\leq 2n-1}$ and $\Sigma^* v \cap L^{\leq 2n-1}$ have not too small a weight under any positive admissible distribution.

We show below a kind of converse: if u is a frequent prefix in L, if v is a frequent suffix in L and if uv is in L, then uv must be frequent.

Lemma 10. *There exists a constant β such that for every non negative integer n, for all strings $u \in \Sigma^{\leq n}$ and $r \in \Sigma^*$, we have*

$$K(u \mid r) \leq \log n + 2 \log \log n - \log \mathbf{m}_r(u\Sigma^*) + \beta$$

and

$$K(u \mid r) \leq \log n + 2 \log \log n - \log \mathbf{m}_r(\Sigma^* u) + \beta$$

Proof: Let $u \in \Sigma^{\leq n}$ and $r \in \Sigma^*$.

We have

$$\mathbf{m}_r(u\Sigma^*) = \alpha_r \sum_{v \in \Sigma^*} 2^{-K(uv \mid r)} \leq c_0 \sum_{v \in \Sigma^*} 2^{-K(uv \mid r)}$$

where c_0 has been defined in Lemma 1.

There exists a constant β_1 such that for all strings u, v, r we have

$$K(\langle u, v \rangle \mid r) \leq K(uv \mid r) + \log n + 2 \log \log n + \beta_1$$

where $|u| \leq n$. Indeed, we only need to know the length of u to compute the pair $\langle u, v \rangle$ from the string uv, and we need a program whose length is at most $\log n + 2 \log \log n$ (up to an additive constant) to compute $|u|$.

A fundamental theorem in Kolmogorov complexity theory (see Symmetry of Algorithmic Information in Li & Vitányi, 1993) says that there exists a constant β_2 such that for all strings u, v, r we have

$$K(\langle u, v \rangle \mid r) \geq K(u \mid r) + K(v \mid u, K(u), r) - \beta_2$$

Therefore,

$$K(uv \mid r) \geq K(u \mid r) + K(v \mid u, K(u), r) - \log n - 2 \log \log n - \beta_1 - \beta_2$$

Then

$$\mathbf{m}_r(u\Sigma^*) \leq c_0 n (\log n)^2 2^{\beta_1 + \beta_2} 2^{-K(u \mid r)} \sum_{v \in \Sigma^*} 2^{-K(v \mid u, K(u), r)}$$

As

$$\sum_{v \in \Sigma^*} 2^{-K(v \mid u, K(u), r)} < 1$$

we have,

$$\mathbf{m}_r(u\Sigma^*) < c_0 n(\log n)^2 2^{\beta_1+\beta_2} 2^{-K(u\,|\,r)}$$

and

$$K(u\,|\,r) \leq \log n + 2\,\log\log n - \log \mathbf{m}_r(u\Sigma^*) + \beta_1 + \beta_2 + \log c_0$$

Let $\beta = \beta_1 + \beta_2 + \log c_0$: we get the result.
The second inequality is showed in a similar way. □

Corollary 3. *There exists a constant c_4 such that for every non negative integer n, for every non empty regular language L with n states, for every $r \in R(L)$ and $a > 0$, and for every pair $(u, v) \in S_r^a$, we have*

$$\mu_r(uv) \geq c_4 \left(\frac{a}{n(\log n)^2} \right)^2$$

Proof: First, there exists a constant γ such that

$$\mu_r(uv) \geq \mathbf{m}_r(uv) \geq 2^{-K(uv\,|\,r)} \geq 2^{-K(u\,|\,r)-K(v\,|\,r)-\gamma}$$

From previous lemma and Lemma 2, we have

$$\begin{aligned} 2^{-K(u\,|\,r)} &\geq \frac{2^{-\beta}\mathbf{m}_r(u\Sigma^*)}{n(\log n)^2} \\ &\geq \frac{2^{-\beta}\mathbf{m}_r(u\Sigma^* \cap L^{\leq 2n-1})}{n(\log n)^2} \\ &\geq \frac{2^{-\beta}\mu_r(u\Sigma^* \cap L^{\leq 2n-1})}{n(\log n)^2}\mathbf{m}_r(L) \\ &\geq c_1 2^{-\beta}\frac{a}{n(\log n)^2} \end{aligned}$$

and in a similar way,

$$2^{-K(v\,|\,r)} \geq c_1 2^{-\beta}\frac{a}{n(\log n)^2}$$

Therefore

$$\mu_r(uv) \geq (c_1)^2 2^{-2\beta-\gamma}\left(\frac{a}{n(\log n)^2}\right)^2$$

Let $c_4 = (c_1)^2 2^{-2\beta-\gamma}$: we get the result. □

Now, we are going to show that if we draw a sufficiently large sample according to a positive admissible distribution it is possible to compute *exactly* a positive teaching set for the target.

We recall some classical results based on the Hoeffding bounds:

Lemma 11. *Let $X_1, \ldots, X_N$ be a sequence of m independent Bernoulli trials, each with probability of success p. Let $S = \frac{X_1+\cdots+X_N}{N}$ be a random variable estimating the parameter p (i.e. $\boldsymbol{E}[S] = p$). Then for $0 \leq \epsilon \leq 1$ the following inequality holds*

$$Pr[|S - p| > \epsilon] \leq 2e^{-2N\epsilon^2}$$

Therefore, if we want to estimate the weight of m events $A_1, \ldots, A_m$ under a probability distribution μ with accuracy ϵ and confidence δ, it is sufficient to draw N elements according to μ where N satisfies $2me^{-2N\epsilon^2} \leq \delta$. Verify that $N = \frac{\ln 2m - \ln \delta}{2\epsilon^2}$ is suitable.

Now, consider the following algorithm:

Algorithm: $\mathcal{T}$

Input: δ, n
Let $a = \frac{c_3}{n(\log n)^2}$ where c_3 is defined in Lemma 8
Let $N_1 = \lceil 9\frac{(n+4)\ln 2 - \ln \delta}{2a^2} \rceil$
suppose that the target language L has at most n states
and that the underlying distribution μ_r is a positive
admissible distribution according to L
Draw N_1 examples according to the oracle $EX(L, \mu_r)$
Discard the examples whose length is greater than $2n - 1$
Build a multisample E with the remaining examples,
i.e. let $w \in \Sigma^*$ and n_w a positive integer
$(w, n_w) \in E$ iff $|w| \leq 2n - 1$ and w has been drawn exactly n_w times
Let $\hat{S}_r = \{(u, v) \in \Sigma^{\leq n} \times \Sigma^{\leq n-1} \mid \exists k > 0\ (uv, k) \in E\}$
Let $\hat{U}_r = \pi_1(\hat{S}_r)$ and $\hat{V}_r = \pi_2(\hat{S}_r)$
we have a sufficiently large sample to estimate the following weights
with good accuracy and confidence
For all $u \in \hat{U}_r$ **compute**
$\hat{\mu}_r(u\Sigma^* \cap L^{\leq 2n-1}) = \Sigma\{n_w \mid (w, n_w) \in E \text{ and } w \in u\Sigma^*\}/N_1$
For all $v \in \hat{V}_r$ **compute**
$\hat{\mu}_r(\Sigma^* v \cap L^{\leq 2n-1}) = \Sigma\{n_w \mid (w, n_w) \in E \text{ and } w \in \Sigma^* v\}/N_1$
Let $\hat{U}_r^{2a/3} = \{u \in \hat{U}_r \mid \hat{\mu}_r(u\Sigma^* \cap L^{\leq 2n-1}) \geq 2a/3\}$ and
$\hat{V}_r^{2a/3} = \{v \in \hat{V}_r \mid \hat{\mu}_r(\Sigma^* v \cap L^{\leq 2n-1}) \geq 2a/3\}$
Let $N_2 = \lceil \frac{1}{c_4}(\frac{3n(\log n)^2}{a})^2 \ln \frac{18n(n+1)}{a^2\delta} \rceil$
Draw N_2 examples according to $EX(L, \mu_r)$
Discard the examples whose length is greater than $2n - 1$
Build a sample E' with the remaining examples
$\hat{S}_r^{2a/3} \leftarrow \{(u, v) \in \hat{U}_r^{2a/3} \times \hat{V}_r^{2a/3} \mid uv \in E'\}$
We will prove that $\hat{S}_r^{2a/3}$ is a positive teaching set with high confidence
Output: $\hat{S}_r^{2a/3}$

Proposition 6. *For every non empty regular language L with n states, for every representation $r \in R(L)$ and for every positive admissible distribution μ_r, the running time of the*

algorithm $\mathcal{T}$ is polynomial in n and $1/\delta$. With a probability greater than $1-\delta$, $\mathcal{T}$ outputs a positive teaching set for L.

Proof: There are less than 2^{n+2} sets of the form $u\Sigma^* \cap L^{\leq 2n-1}$ and $\Sigma^* v \cap L^{\leq 2n-1}$ where $u, v \in \Sigma^{\leq n}$. Then, from Lemma 11, with a confidence greater than $1-\delta/2$, we have for all $u \in \hat{U}_r$

$$\left|\hat{\mu}_r(u\Sigma^* \cap L^{\leq 2n-1}) - \mu_r(u\Sigma^* \cap L^{\leq 2n-1})\right| \leq \frac{a}{3}$$

and for all $v \in \hat{V}_r$

$$\left|\hat{\mu}_r(\Sigma^* v \cap L^{\leq 2n-1}) - \mu_r(\Sigma^* v \cap L^{\leq 2n-1})\right| \leq \frac{a}{3}$$

Therefore, with a probability greater than $1-\delta/2$, we have:

$$U_r^a \subseteq \hat{U}_r^{2a/3} \subseteq U_r^{a/3} \quad \text{and} \quad V_r^a \subseteq \hat{V}_r^{2a/3} \subseteq V_r^{a/3}$$

hence,

$$S_r^a \subseteq S = \left\{(u, v) \in \hat{U}_r^{2a/3} \times \hat{V}_r^{2a/3} \,\middle|\, uv \in L\right\} \subseteq S_r^{a/3}$$

Suppose now that these inequalities are satisfied. We will prove that $\hat{S}_r^{2a/3} = S$ with a probability greater than $1-\delta/2$.

Let $(u, v) \in S$. From Corollary 3, we have $\mu_r(uv) \geq c_4(\frac{a}{3n(\log n)^2})^2$. From Lemma 9, we have $card(S) \leq 9n(n+1)/a^2$.

The number N_2 has been chosen in such a way that,

$$\frac{9n(n+1)}{a^2}\left(1 - c_4\left(\frac{a}{3n(\log n)^2}\right)^2\right)^{N_2} \leq \frac{9n(n+1)}{a^2} e^{-N_2 c_4\left(\frac{a}{3n(\log n)^2}\right)^2} \leq \delta/2$$

i.e., with a probability greater than $1-\delta/2$, all strings of $\{uv \mid (u, v) \in S\}$ will be drawn and the algorithm will output S. As from Lemma 4, S is a positive teaching set, we get the result.

The algorithm is clearly polynomial in n and $1/\delta$. More precisely, verify that for every $\alpha > 0$, the number of examples drawn is $O(n^{4+\alpha} \ln(1/\delta))$ and the running time is $O(n^{5+\alpha} \ln(1/\delta))$. □

Theorem 2. *The class of regular languages is probably exactly learnable from simple positive examples.*

Proof: Consider the following algorithm:

Learning Algorithm:

Input: δ, l
Let us take l as an upper bound for the number of states of the target
Run algorithm $\mathcal{T}$ with inputs δ, l
Let S be the output
Run algorithm $\mathcal{A}$ with input S
Let r' be the output
Output: r'

Propositions 6 and 4 show that, for every regular language L such that $size(L) \leq l$, and every representation $r \in R(L)$, the previous algorithm $\mathcal{A}(\mathcal{T}(l, \delta))$ outputs a representation of L with probability greater than $1 - \delta$, when it is feeded with the oracle $EX(L, \mu_r)$. □

Algorithm $\mathcal{T}$ runs in time $O(l^{5+\alpha} \ln(1/\delta))$ for every $\alpha > 0$. As $S \subseteq S_r^{a/3}$ and from Lemma 9, $Card(\pi_1(S))$ and $Card(\pi_2(S))$ are in $O(l/a)$ i.e. in $O(l^{2+\alpha})$ for every $\alpha > 0$. The length of the longest element in $\pi_1(S)$ and $\pi_2(S)$ is $\leq l$. Algorithm $\mathcal{A}$ can be implemented in time $O(Card(\pi_1(S))Card(\pi_2(S))Max\{|u|/u \in \pi_1(S) \cup \pi_2(S)\})$ that is $O(l^{5+\alpha})$ for every $\alpha > 0$. Consequently, the total running time of the learning algorithm is in $O(l^{5+\alpha} \ln(1/\delta))$ for every $\alpha > 0$.

6.1. Miscellaneous remarks

– It is important to note that our result is independent on the reference Turing machine which has been chosen to define the Kolmogorov complexity. Indeed, the choice of a particular Turing machine U has never been mentioned or used.
– We still don't have any rigorous definition of what collusion is. Therefore, we cannot prove that no collusion phenomenon can occur in learning regular languages from simple positive examples. But whatever a precise definition would be, if the target language could be encoded by positive examples and if the learner could use this information, it is likely that the class of regular language would be learnable in the model of Goldman and Mathias. That is, the fact that REG is not GM learnable from positive examples is maybe a sufficient reason to think that all danger of collusion is avoided.
– Lastly, note that our algorithm uses the general heuristic principle stated in Section 3: each pair $(u, v) \in \hat{U}_r^{2a/3} \times \hat{V}_r^{2a/3}$ constitutes a (micro) current hypothesis which predicts that the string uv is a simple positive string. If uv does not appear in a reasonably large new sample, the learning algorithm discards this hypothesis.
– It would have been nice to get rid of l as it is possible in classical PAC framework (Haussler et al., 1991). Unfortunately, that seems impossible. Suppose, for example, that we want to differenciate Σ^* and $\Sigma^{\leq N}$ where N is greater than the available running time allowed to learn Σ^*. No efficient strategy seems available. And as there exist very long strings with pretty small complexity, turning the exact learning requirement into an approximative one does not help. Maybe, it should have been fairer to say that for every integer l, the class REG^l is probably exactly learnable.

To conclude, let us study how the previous result can be adapted to the simple PAC model of Li and Vitányi (1991).

As they do in Li and Vitányi (1991) with k-reversible languages, we define a notion of *simple regular language*.

Definition 11. Let L be a non empty regular language and let $A = (\Sigma, Q, q_0, T, \delta)$ be a trimmed minimal automaton which recognizes L. Let c be an integer and $n = Card(Q)$. We say that L is c-simple if for every state $q \in Q$,

$$K(u_q \mid r_L) \leq c \log n \quad \text{and} \quad K(v_q \mid r_L) \leq c \log n$$

Proposition 7. *The class of c-simple regular languages is learnable from positive simple examples provided that the examples are drawn according to the* (*positive restriction of*) *Solomonoff-Levin distribution* **m**.

Proof: The only place where we used the fact that the examples are drawn according to μ_r *where r is a representation of the target language* is Lemma 7. But, if the target is c-simple, we get a similar conclusion when we use non conditional complexity ($K(\cdot)$ rather that $K(\cdot \mid r)$). All the following lemmas can be adapted to this case in a straightforward way. □

A regular language L is 0-reversible if the mirror automaton of the minimal trimmed DFA A that recognizes L is deterministic. A 0-reversible language is simple if for every state q, the least string u such that $\delta(q_0, u) = q$ and the least string v such that $\delta(q, v) \in T$ are simple. Li and Vitanyi have proved in Li and Vitányi (1991) that simple 0-reversible languages are PAC learnable from **m**.

But, if L is 0-reversible, we have

$$u_1 v \in L \quad \text{and} \quad u_2 v \in L \Rightarrow \delta(q_0, u_1) = \delta(q_0, u_2)$$

Hence, for every state q, v_q is the least string such that $\delta(q, v_q) \in T$. We see that the notion of *simple regular language* generalizes the notion of *simple* 0*-reversible language*. A similar remark can be made about k-reversible languages. Therefore, the Proposition 7 generalizes the results of Li and Vitányi (1991) mentioned above.

7. Conclusion

Learning from positive data is an important topic in Computational Learning Theory, and specially in the domain of Grammatical Inference since it is believed that it is possible to learn the syntax of a natural language solely from positive instances. However, classical models provide no tools to overcome the main difficulty in this kind of learning, i.e. to avoid overgeneralization. This is specially true when we want the learning to be efficient, i.e. polynomial in the size of the target representation. In the field of learning DFAs, most of the positive results rely on an extra-hypothesis concerning the distribution of the examples

to the learner: they must contain a "teaching set" as for the class of k-reversible languages (Angluin, 1982) or they must be structured (Sakakibara, 1992).

Here, we use a generalization of the learning model of Li and Vitányi (Li & Vitányi, 1991; Denis, D'Halluin & Gilleron, 1996) that makes the hypothesis that simple examples are more frequent than complex ones, using the notion of Kolmogorov complexity to define what is a simple example.

We proved that the class of regular languages is probably exactly learnable in this model.

Beyond the technical aspects of this result, we would like to emphasize the fact that the hypothesis made by the model and the heuristic used by the learning algorithm have some relevance from a cognitive point of view.

- The simplicity of an example or the fact that it is characteristic of a concept is often used, without being defined. Kolmogorov complexity allows to give a rigorous definition of this intuitive notion,
- Supposing that simple examples are more frequent is a plausible hypothesis in numerous natural learning contexts,
- Supposing that a current hypothesis may be ruled out if some simple expected events do not occur seems to be a plausible heuristic too,
- Lastly, the sole property of the model that we use to show the learnability of the class of regular languages is the following: there exists a subset S of the language, composed of frequent words, containing sufficiently rich information to allow the reconstruction of the language and such that if a correct word has components appearing in S, then it must be in S too. Again, we think that this hypothesis is plausible from a linguistic point of view.

As a future work, we would like to develop this result in both theoretical and experimental directions. The class of regular languages is too poor to describe significant fragments of natural language—context free languages are needed at least. What complexity levels are reachable by the model and the techniques developed here?

The Solomonoff-Levin distribution is not computable and it would be interesting to isolate which properties are needed in order to keep this result. This would allow to design more practicable families of admissible distributions. We would like too to collect real data from natural language corpuses and study to what extent it is possible to suppose they have been generated by such admissible distributions.

References

Angluin, D. (1980). Inductive inference of formal languages from positive data. *Inform. Control, 45:2*, 117–135.

Angluin, D. (1982). Inference of reversible languages. *J. ACM, 29:3*, 741–765.

Angluin, D. (1987). Learning regular sets from queries and counterexamples. *Information and Computation, 75:2*, 87–106.

Carrasco, R., & Oncina, J. (1994). Learning stochastic regular grammars by means of a state merging method. In *International Conference on Grammatical Inference* (pp. 139–152), Heidelberg: Springer-Verlag.

Castro, J., & Guijarro, D. (1998). Query, pacs and simple-pac learning. Technical report, Dept. Llenguatges i Sistemes Informàtics.

de Jongh, D, & Kanazawa, M. (1996). Angluin's theorem for indexed families of r.e. sets and applications. In *Proc. 9th Annu. Conf. on Comput. Learning Theory* (pp. 193–204), New York, NY: ACM Press.

Denis, F. (1998). PAC learning from positive statistical queries. In M. M. Richter, C. H. Smith, R. Wiehagen, & T. Zeugmann, (Eds.), *Proceedings of the 9th International Conference on Algorithmic Learning Theory (ALT-98)* (pp. 112–126), Berlin: Springer Vol. 1501 of *LNAI*.

Denis, F., D'Halluin, C., & Gilleron, R. (1996). PAC learning with simple examples. In *13th Annual Symposium on Theoretical Aspects of Computer Science* (pp. 231–242), Grenoble, France: Springer. Volume 1046 of *INCS*.

Denis, F., & Gilleron, R. (1997a). PAC learning under helpful distributions. In M. Li, & A. Maruoka, (Eds.), *Proceedings of the 8th International Workshop on Algorithmic Learning Theory (ALT-97)* (pp. 132–145), Berlin: Springer. Vol. 1316 of *LNAI*.

Denis, F., & Gilleron, R. (1997b). Pac learning under helpful distributions. Submitted. Long version available at ftp://www.grappa.univlille3.fr/pub/reports/helpful.ps.

D'Halluin, C. (1998). Apprentissage par exemples simples; plate-forme d'apprentissage de langages réguliers. PhD thesis, Université Lille 1.

Gold, E. (1967). Language identification in the limit. *Inform. Control, 10*, 447–474.

Gold, E. (1978). Complexity of automaton identification from given data. *Inform. Control, 37*, 302–320.

Goldman, S. A., & Mathias, H. D. (1996). Teaching a smarter learner. *Journal of Computer and System Sciences, 522*, 255–267.

Haussler, D., Kearns, M., Littlestone, N., & Warmuth, M. K. (1991). Equivalence of models for polynomial learnability. *Inform. Comput., 95:2*, 129–161.

Head, T., Kobayashi, S., & Yokomori, T. (1998). Locality, reversibility and beyond: Learning languages from positive data. In *ALT 98, 9th International Conference on Algorithmic Learning Theory* (pp. 191–204), Springer-Verlag. Vol. 1501 of *Lecture Notes in Artificial Intelligence*.

Higuera, C. D. L. (1997). Characteristic sets for polynomial grammatical inference. *Machine Learning, 27*, 125–137.

Kanazawa, M. (1996). Identification in the limit of categorial grammars. *Journal of Logic, Language, and Information, 5:2*, 115–155.

Kearns, M., & Valiant, L. (1994). Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM, 41:1*, 67–95.

Kearns, M. J., & Vazirani, U. V. (1994). *An Introduction to Computational Learning Theory*. MIT Press.

Koshiba, T., & E. Mäkinen, Y. T. (1997). Learning deterministic even linear languages from positive examples. *Theoretical Computer Science, 185*, 63–79.

Li, M., & Vitányi, P. (1991). Learning simple concepts under simple distributions. *SIAM J. Comput., 20*, 911–935.

Li, M., & Vitanyi, P. (1993). *An Introduction to Kolmogorov Complexity and Its Applications*. Text and Monographs in Computer Science. Springer-Verlag.

Natarajan, B. (1991a). *Machine Learning: A Theoretical Approach*. San Mateo, CA: Morgan Kaufmann.

Natarajan, B. K. (1991b). Probably approximate learning of sets and functions. *SIAM J. COMPUT., 20:2*, 328–351.

Oncina, J., & Garcia, P. (1992). Inferring regular languages in polynomial update time. In *Pattern Recognition and Image Analysis*, pp. 49–61.

Parekh, R., & Honavar, V. (1997). Learning DFA from simple examples. In M. Li, & A. Maruoka, (Eds.), *Proceedings of the 8th International Workshop on Algorithmic Learning Theory (ALT-97)* (pp. 16–131), Berlin: Springer. Vol. 1316 of *LNAI*.

Pitt, L. (1989). Inductive inference, DFAs, and computational complexity. In *Proceedings of AII-89 Workshop on Analogical and Inductive Inference*. (pp. 18–44), Heidelberg: Springer-Verlag. Vol. 397 of Lecture Notes in Artificial Intelligence.

Rabiner, L. R., & Juang, B. H. (1986). An introduction to hidden markov models. *IEEE ASSP Magazine, 3:1*, 4–16.

Sakakibara, Y. (1992). Efficient learning of context-free grammars from positive structural examples. *Information and Computation, 97:1*, 23–60.

Sakakibara, Y. (1997). Recent advances of grammatical inference. *Theoretical Computer Science, 185:1*, 15–45.

Shvayster, H. (1990). A necessary condition for learning from positive examples. *Machine Learning, 5*, 101–113.

Stolcke, A., & Omohundro, S. (1994). Inducing probabilistic grammars by Bayesian model merging. *Lecture Notes in Computer Science, 862*, 106–118.

Tellier, I. (1998). Meaning helps learning syntax. In *Proceedings of ICGI'98 Workshop on Grammatical Inference* (pp. 25–36), Vol. 1433 of *Lecture Notes in Artificial Intelligence*.

Valiant, L. (1984). A theory of the learnable. *Commun. ACM, 27:11*, 1134–1142.

Yokomori (1995). On polynomial-time learnability in the limit of strictly deterministic automata. *Machine Learning, 2*, 153–179.

Received December 1, 1998
Revised January 14, 2000
Accepted March 30, 2000
Final manuscript May 12, 2000