PAC LEARNING UNDER HELPFUL DISTRIBUTIONS*,**

François Denis¹ and Rémi Gilleron¹

Abstract. A PAC teaching model – under helpful distributions – is proposed which introduces the classical ideas of teaching models within the PAC setting: a polynomial-sized teaching set is associated with each target concept; the criterion of success is PAC identification; an additional parameter, namely the inverse of the minimum probability assigned to any example in the teaching set, is associated with each distribution; the learning algorithm running time takes this new parameter into account. An Occam razor theorem and its converse are proved. Some classical classes of boolean functions, such as Decision Lists, DNF and CNF formulas are proved learnable in this model. Comparisons with other teaching models are made: learnability in the Goldman and Mathias model implies PAC learnability under helpful distributions. Note that Decision lists and DNF are not known to be learnable in the Goldman and Mathias model. A new simple PAC model, where "simple" refers to Kolmogorov complexity, is introduced. We show that most learnability results obtained within previously defined simple PAC models can be simply derived from more general results in our

Mathematics Subject Classification. 68Q30, 68Q32.

Introduction

Many concept classes are not efficiently learnable in the basic PAC learning model of [34]. We think that the main reason of this fact is due to the *distribution*

Keywords and phrases: PAC learning, teaching model, Kolmogorov complexity.

^{*} This research was partially supported by "Motricité et Cognition": Contrat par objectifs région Nord/Pas-de-Calais.

^{**} A preliminary version was published in the Proceedings of the 8th International Workshop on Algorithmic Learning Theory ALT'97 [11].

¹ Équipe GRAPPA, LIFL UPRESA 8022 du CNRS, Université de Lille 1, 59655 Villeneuved'Ascq, France; e-mail: denis@lifl.fr & gilleron@lifl.fr

free requirement, that the learning algorithm must work with respect to an arbitrary distribution, even if the same distribution is used to provide examples to the learner and to evaluate the output hypothesis. An argument in support of this claim can be found in the Shapire result [32] on equivalence of weak and strong learning: even an efficient algorithm whose hypotheses are only sligthly better than "random guessing" can be used to obtain an efficient PAC algorithm. The proof of this rather surprising result relies on the existence of a weak hypothesis whatever distribution is used to provide the examples.

In many practical learning situations, it is likely that the data are "representative" of the target concept rather than random or arbitrary. Therefore, it is natural to investigate learning methods that assume that the source of examples is "helpful". There are several ways to assume that examples are not arbitrary: the learner may ask queries (see [2] for an overview); the class of distributions used to draw examples can be restricted, thus the learning algorithm knows something about the underlying distribution [3, 10, 18, 20, 24]; a teaching set may be designed in order to help the learner [13, 14, 33]. For instance, Goldman and Mathias assume that a teacher builds a teaching set related to the target concept and that an adversary adds to this set as many new examples as he wants. Then, the learner must identify exactly the target from this sample.

In the present paper, a new PAC learning model with a helpful source of examples is proposed. It is supposed that the teacher knows a representation of the function to be teached and uses this representation to define a polynomial size teaching set (the useful examples). Then, a distribution is said to be helpful if each useful example has a non-zero weight. The class of admissible distributions is restricted to the class of helpful distributions. So, the class of admissible distributions both depends on the target concept and on the representation the teacher has in mind. Then, we want to give more time to the learning algorithm when the useful examples are badly represented by the underlying distribution. In order to do that, we allow the learning algorithm to run in time polynomial in $1/D_{\min}(c)$ (together with the usual parameters) where $D_{\min}(c)$ is the least weight of a useful example of the target concept c. If the least weight of a useful example is quite small, then the parameter $1/D_{\min}(c)$ could be quite large. For instance, when the uniform distribution is used, exponential time is allowed and learning becomes trivial. Therefore, our model may at first seem too permissive. But, the reader should note that we meet the usual time requirements when the underlying distribution D is really helpful, i.e. when $1/D_{\min}(c)$ is polynomial in the size of the target concept c. Hopefully, in many practical learning situations, the underlying distribution satisfies such a condition. Therefore, our model can be seen as a restriction of the standard model of Valiant to a large and meaningful class of

In PAC learning theory, the Occam's razor theorem [4] is one of the most important results. Let us recall that an Occam algorithm is an algorithm that must find a short hypothesis consistent with the observed data. Compress the data in order to learn is a classical heuristic. The Occam's razor theorem states that any efficient Occam algorithm is also a PAC learning algorithm. This theorem provides

both a formal justification of the Occam principle and an empirical justification of the PAC learning model. In the PAC model under helpful distributions, the runtime of a learning algorithm depends on the least weight of a useful example. In our definition of an Occam algorithm, the size condition for the output hypothesis depends on the least frequency of a useful example in the input sample. We prove an Occam's razor theorem in the PAC learning model under helpful distributions. Moreover, as in the usual PAC setting [5], we prove a converse of the Occam's razor theorem for many natural classes. In our opinion, the Occam's razor theorem and its converse provide a theoretical validation of our model.

Using our Occam's razor theorem, we prove that decision lists are PAC learnable under helpful distributions. The Occam algorithm for decision lists examines the examples by decreasing multiplicity. This corresponds to a usual heuristic in practical learning algorithms. Note that most usual PAC learning algorithms do not really take the multiplicity of examples into account. We also prove that DNF and CNF formulae are learnable under helpful distributions.

Other teaching models were previously defined in the framework of exact learning. For instance, we consider the teaching model of [14] and compare it with our model. Learnability in the model of Goldman and Mathias implies PAC learnability under helpful distributions. The converse is probably false as exact identification is more difficult to achieve than approximate learning, but this is not proved. For example, the class of decision lists is PAC learnable under helpful distributions but is not known to be learnable in the Goldman and Mathias model.

Li and Vitányi introduced Kolmogorov complexity in PAC learning setting in [20]. A string is said to be simple if it has a low Kolmogorov complexity. They showed that simple concepts can be learned from simple examples. This approach has been carried on in several papers [7, 10]. We define here a new simple PAC learning model. A teacher is said to be simple if for each concept c, useful examples are of low conditional Kolmogorov complexity relatively to c. A concept class is said to be PAC learnable with simple teacher if it is PAC learnable under helpful distributions for some simple teacher. It is easy to prove that any computable teacher is a simple teacher. Therefore, the classes of decision lists, DNF and CNF are PAC learnable with a simple teacher. We compare our model with previously defined simple PAC learning models. As a corollary, we prove that $\log n$ -decision lists are simple PAC learnable in the sense of Li and Vitányi and our learning algorithm is based on an Occam algorithm. This last result was first proved in [7], where the possibility of using an Occam algorithm was discussed.

Our model is defined in Section 1. The Occam's razor theorem and its converse are given and proved in Section 2. Learnability of decision lists is proved in Section 3. We compare our model with previously defined teaching models in Section 4. The simple PAC Learning models are defined and compared in Section 5.

1. PAC LEARNING UNDER HELPFUL DISTRIBUTIONS

1.1. NOTATIONS AND PRELIMINARY DEFINITIONS

We assume familiarity with the basic facts about PAC learning theory, see for example [19].

Let \mathcal{B}_n be the set of boolean functions from $X_n = \{0,1\}^n$ into $\{0,1\}$. Let $\mathcal{B} = \cup_{n \geq 1} \mathcal{B}_n$ and $X = \cup_{n \geq 1} X_n$. A class \mathcal{F} of boolean functions is a subset of \mathcal{B} . A representation scheme for a class of boolean functions \mathcal{F} is a function $R: \mathcal{F} \to 2^{\Sigma^*}$ where Σ is a finite alphabet and such that for each f and f' in \mathcal{F} , R(f) is not empty and if $f \neq f'$, $R(f) \cap R(f') = \emptyset$. We suppose that R is computable in polynomial-time, that is, there exists a polynomial-time deterministic algorithm which takes as input a pair of strings x and c and outputs f(x) if $c \in R(f)$ and error otherwise. A concept class C is defined by $C = \cup_{f \in \mathcal{F}} R(f)$ for a given class of function \mathcal{F} . We will often identify a concept c in C and the function f which is represented by c. When two concepts c and c' represent the same function, we note $c \equiv c'$. We define the size of a concept c as its length |c| and we suppose for convenience that $|c| \geq n$.

Note that in the standard PAC learning model, a representation scheme being given, a concept may have several names while in our model, each name defines one concept. For example, if we consider the DNF representation scheme of boolean functions, x_1 and $x_1\overline{x}_2 + x_1x_2x_3 + x_1x_2\overline{x}_3$ are two concepts which describe the same function. In this way, different sets of useful examples can be associated with x_1 and $x_1\overline{x}_2 + x_1x_2x_3 + x_1x_2\overline{x}_3$.

An example of a concept c is a pair (x, c(x)), where x is in the domain of c. An example (x, c(x)) is positive if c(x) = 1 and negative otherwise. We denote by EX(c) (respectively POS(c), NEG(c)) the set of all examples (respectively positive examples, negative examples) of a concept c. A sample S of c is a multiset of examples of c. Let c be a concept over X_n and let S be a sample of c. We denote by S(x, c(x)) the number of occurrences of (x, c(x)) in S. We let $Card(S) = \sum_{x \in X_n} S(x, c(x))$ and ||S|| = nCard(S). Let c be a target concept over X_n and let D be any fixed probability distribution over X_n . Let EX(c, D) be a procedure that runs in unit time and that at each call returns an example (x, c(x)), where x is drawn randomly and independently according to x. If x is any concept in x over x, we define x or x is inconsistent with an example of x which is drawn randomly according to x, x, x is inconsistent with an example of x which is drawn randomly according to x.

$$error(c') = D(\lbrace x | c(x) \neq c'(x) \rbrace) = D(c\Delta c').$$

1.2. Definition of our model

Let C be a concept class over X. A teaching set for $c \in C$ is a subset of EX(c). A teacher for C is a mapping \mathcal{T} which associates with each concept c, a teaching set $\mathcal{T}(c)$, and such that there is a constant k such that for every concept c, $Card(\mathcal{T}(c)) \leq |c|^k$.

Let C be a concept class and let \mathcal{T} be a teacher for C. Let c be a target concept over X_n and let D be any fixed probability distribution over X_n . Let us define

$$D_{\min}(c) = \begin{cases} \min\{D(x) \mid (x, c(x)) \in \mathcal{T}(c)\} & \text{if } \mathcal{T}(c) \neq \emptyset, \\ 1 & \text{otherwise.} \end{cases}$$
 (1)

A distribution D is said to be helpful w.r.t. c and \mathcal{T} if $D_{\min}(c) \neq 0$, i.e. each useful example has a non-zero probability to be drawn according to D. A distribution D can be helpful w.r.t. c and be not helpful w.r.t. c'. Therefore the set of helpful distributions depends on the target concept. But the reader should also note that a distribution which is not null on the instance space is helpful for every target concept.

We now define PAC learnability under helpful distributions.

Definition 1.1. Let C be a concept class.

- Given a teacher \mathcal{T} for C, an algorithm \mathcal{A} is a PAC learning algorithm for C under helpful distributions if \mathcal{A} takes as input $\epsilon \in (0,1]$, $\delta \in (0,1]$, an integer l, and for all concepts c in C with $|c| \leq l$ and all helpful probability distributions D, \mathcal{A} is given access to EX(c,D) and \mathcal{A} outputs some h in C, such that with probability at least 1δ , $error(h) \leq \epsilon$.
- C is PAC learnable under helpful distributions if there are a teacher \mathcal{T} for C and a PAC learning algorithm \mathcal{A} for C under helpful distributions which runs in time polynomial in $1/\epsilon$, $1/\delta$, l, and $1/D_{\min}(c)$.

Our model generalizes the PAC learning model because a concept class C is PAC learnable if and only if C is PAC learnable with the empty teacher (for each concept c, the teaching set is empty). There are two differences between our model and the usual PAC model. First, the model is not a fixed distribution set model because the set of helpful distributions depends on the target concept. The learner has only to succeed for helpful distributions. But it is easy to verify that alone, this modification does not increase the learning ability of the model. Second, we require that a learning algorithm runs in time polynomial in $1/D_{\min}(c)$. But, note that if the parameter $1/D_{\min}(c)$ is polynomial in |c|, the time requirement is the same as in the usual PAC setting. Distributions D such that $1/D_{\min}(c)$ is polynomial in |c| could be described as really helpful but note that the parameter $D_{\min}(c)$ is not known from the learner.

The halting criterion of the learning algorithm may be probabilistic:

Definition 1.2. Let C be a concept class and let \mathcal{T} be a teacher. C is PAC learnable under helpful distributions in usually polynomial time if there is a PAC learning algorithm \mathcal{A} for C under helpful distributions such that, with probability at least $1 - \delta$, \mathcal{A} halts in time polynomial in $1/\epsilon$, $1/\delta$, l, and $1/D_{\min}(c)$.

We have assumed that an upper bound l of the size of the target is an input of the learning algorithm. In the PAC's framework, when l is unknown, the learning algorithm uses a hypothesis testing algorithm for the halting criterion [16]. This technique can easily be adapted to our model. We have: if $\mathcal{A}(\epsilon, \delta, l)$ is a usually

polynomial time PAC learning algorithm for C under helpful distributions, there exists a usually polynomial time PAC learning algorithm \mathcal{A}' which only takes as input ϵ and δ .

2. Occam's razor theorem

An algorithm is an Occam algorithm if it finds a short hypothesis consistent with the observed data. The Occam's razor theorem [4] states that any efficient Occam algorithm is also a PAC learning algorithm.

We slightly modify the succinctness condition for an algorithm to be an Occam algorithm adding the parameter $1/f_{\min}(S,c)$, where $f_{\min}(S,c)$ is the smallest frequency of a useful example in the sample S. We prove an Occam's razor theorem in our model. Moreover, like in the usual PAC setting, we prove a converse of this theorem for concept classes which are strongly closed under exceptions. These two theorems confirm, from a theoretical point of view, the relevance of our model.

Let C be a concept class and \mathcal{T} be a teacher for C. Let S be a non empty sample of some concept c in C. Let us define

$$f_{\min}(S, c) = \begin{cases} \min \left\{ \frac{S(x, c(x))}{Card(S)} \mid (x, c(x)) \in \mathcal{T}(c) \right\} & \text{if } \mathcal{T}(c) \neq \emptyset, \\ 1 & \text{otherwise} \end{cases}$$

where S(x, c(x)) denotes the number of occurrences of (x, c(x)) in S. $f_{\min}(S, c)$ is the least frequency of a useful example in the multiset S.

Note that if D is the uniform distribution on S, we have $D_{\min}(c) = f_{\min}(S,c)$. Note also that if $\mathcal{T}(c) \subseteq S$, then $1/f_{\min}(S,c) \leq Card(S)$. Lastly, if D is a really helpful distribution for c (i.e. $D_{\min}(c)$ is polynomial in 1/|c|), with a high confidence, a polynomial size sample S drawn according to D will be such that $f_{\min}(S,c)$ is polynomial in 1/|c| too.

Definition 2.1. Let C be a concept class and \mathcal{T} be a teacher for C.

B is an Occam algorithm for C, \mathcal{T} if there exists constants $a \geq 0$, $b \geq 0$ and $0 \leq \alpha < 1$ such that with a sample S of c in X_n such that $\mathcal{T}(c) \subseteq S$ on input, B outputs a hypothesis concept c' such that:

- c' is consistent with S;
- $|c'| \le a(|c|/f_{\min}(S,c))^b (Card(S))^{\alpha};$
- B runs in polynomial time in |c|, Card(S).

Let us point out the differences between the definition of an Occam algorithm in Valiant's PAC learning model and the definition of an Occam algorithm in our model. First, an Occam algorithm in our model only has to work properly if the sample S contains the teaching set $\mathcal{T}(c)$ of the target concept c. Second, c' needs to be short only if $f_{\min}(S,c)$ is not too small. If $f_{\min}(S,c)$ is polynomial in 1/|c|, we meet the usual size requirement.

We can now give our main theorem:

Theorem 2.2. Let C be a concept class and let \mathcal{T} be a teacher for C. If there is an Occam algorithm for C, \mathcal{T} , then C is PAC learnable under helpful distributions in usually polynomial time.

The proof can be found in Appendix A. The differences with the classical proof are twofold:

- given an Occam algorithm for C, \mathcal{T} , a PAC learning algorithm \mathcal{A} for C under helpful distributions iterates over larger guesses for $1/D_{\min}(c)$;
- consequently, the halting criterion of the PAC learning algorithm is probabilistic. The halting criterion contains a hypothesis testing algorithm and a consistency test. These two tests run in polynomial time.

As in the usual PAC setting, a converse of the Occam theorem holds for concept classes which are strongly closed under exception. A concept class is closed under exceptions if incorporating the exceptions of a finite set in a concept of the class results in a concept also in the concept class. This property is useful when it is efficiently computable. This leads to the notion of strong closure under exception which we now formally define:

Definition 2.3 ([5], see also [23]). A concept class C is strongly closed under exception if there exists an algorithm A and constants α and β such that

- (i) A takes as input a concept $c \in C$ and a finite sample S of c and outputs a concept c' in C such that for all $(x, c(x)) \in S$, $c'(x) \neq c(x)$ and for all $(x, c(x)) \notin S$, c'(x) = c(x);
- (ii)

$$|c'| \le \alpha(|c| + ||S||) \log(|c| + ||S||) + \beta;$$

(iii) A runs in time polynomial in the length of its input.

The class of DNF formulas is strongly closed under exception. The class of k-DNF is not strongly closed under exception.

We are ready to give the converse of Theorem 2.2.

Theorem 2.4. Let C be a concept class and let \mathcal{T} be a polynomial teacher for C. If C is PAC learnable under helpful distributions in usually polynomial time and if C is strongly closed under exception then there exists a randomized algorithm B such that with δ and a sample S of C such that $\mathcal{T}(C) \subseteq S$ on input, then with probability at least $1 - \delta$, B outputs a hypothesis concept C such that:

- c' is consistent with S;
- there exists constants $a \ge 0, b \ge 0$ and $\alpha < 1$ which do not depend on S and c such that $|c'| \le a(|c|/f_{\min}(S,c))^b(card(S))^{\alpha}$;
- \mathcal{B} runs in polynomial time in $\log(1/\delta)$, Card(S) and |c|.

The proof can be found in Appendix B.

3. Learning decision lists

A decision list over x_1, \ldots, x_n is an ordered sequence

$$L = (m_1, b_1), \ldots, (m_p, b_p)$$

of terms, in which each m_i is a monomial over x_1, \ldots, x_n , and each $b_i \in \{0, 1\}$. The last monomial is always $m_p = 1$. For any input $a \in \{0, 1\}^n$, the value L(a) is defined as b_i , where i is the smallest index satisfying $m_i(a) = 1$.

The class of k-decision lists (each monomial contains at most k literals) is PAC learnable [29]. Decision lists are not known to be PAC learnable, but lower bounds on learning decision lists are given in [15]. We prove in this section that the concept class of decision lists is PAC learnable under helpful distributions. Note that decision lists are a superset of DNF formulas and thus DNF formulas are learnable using decision lists. It can be proved, using a greedy heuristic, that DNF and CNF formulas are properly learnable under helpful distributions.

First, let us define the teacher \mathcal{T} [20]. Let m be a monomial over x_1, \ldots, x_n . Let $0_m \in X_n$ be defined by $m(0_m) = 1$ and vector 0_m has the value 0 for variables not in m. Let $1_m \in X_n$ be defined by $m(1_m) = 1$ and vector 1_m has the value 1 for variables not in m. Let c be a representation of a decision list $L = (m_1, b_1), \ldots, (m_p, b_p)$, we define

$$\mathcal{T}(c) = \{(0_{m_i}, c(0_{m_i})) \mid 1 \le i \le p\} \cup \{(1_{m_i}, c(1_{m_i})) \mid 1 \le i \le p\}.$$

The purpose of the teaching set for a target decision list c is to allow the construction of the monomials of c. Indeed, let x, x' in X_n , we define $x \diamond x'$ as the monomial over x_1, \ldots, x_n which contains x_i if x and x' have "1" in position i, contains $\overline{x_i}$ if x and x' have "0" in position i, and does not contain variable x_i otherwise $(1 \le i \le n)$. Given the teaching set $\mathcal{T}(c)$ of some decision list c, the set $\{m = x \diamond x' \mid (x, c(x)), (x', c(x')) \in \mathcal{T}(c)\}$ contains the monomials of c.

Proposition 3.1. Decision lists are PAC learnable under helpful distributions.

Proof. We denote by DL the class of decision lists. We give an Occam algorithm for DL, \mathcal{T} which is based on the Rivest's algorithm for k-decision lists, and then we use our Occam's razor theorem. Our algorithm (see Fig. 1) uses examples in S by decreasing number of occurrences until we are ensured that all examples in the teaching set have been used.

It is now easy to prove that OccamDL is an Occam algorithm for DL, \mathcal{T} . Let c be the target decision list. Let us suppose that $\mathcal{T}(c) \subseteq S$, and let $j = \lceil 1/f_{\min}(S,c) \rceil$. OccamDL halts at most at step i=j because $\mathcal{T}(c) \subseteq S_j$, thus the set M_j contains all the monomials of c. The hypothesis concept c' is consistent with S and the length of c' is bounded by $Card(M_j) \leq (\lceil 1/f_{\min}(S,c) \rceil)^2$. As $1/f_{\min}(S,c) \leq Card(S)$, it is easy to prove that OccamDL runs in polynomial time in |c|, Card(S). Now, Theorem 2.2 can be applied.

```
Algorithm OccamDL
input: S
begin
      Set i to 1; set hypothesis concept c' to the empty list; set S' to S
      while S' \neq \emptyset
           Set S_i to \{(x, c(x)) \in S \mid S(x, c(x)) \ge \frac{Card(S)}{i}\}
           Set M_i to \{m = x \diamond x' \mid (x, c(x)) \in S_i, (x', c(x')) \in S_i\}
           while there is a monomial m in M_i satisfied by an example (y, b)
           in S' and by no example (z, \overline{b}) in S'
                Set c' to c' + (m, b)
                Set S' to S' \setminus \{(y, b) \in S \mid m(y) = 1\}
           endwhile
          Set i to i+1
      endwhile
      output hypothesis concept c'
end
```

Figure 1. An Occam algorithm for decision lists.

4. Teaching models

In the context of automaton identification from finite data, Gold has defined a learning model based on the idea of "representative sets" [12]. We reformulate this model with our notations and we only consider fixed-length examples.

Definition 4.1. A concept class C is *learnable from given data* if there exists a teacher \mathcal{T} , a polynomial p and an algorithm \mathcal{A} such that:

- for all concepts c in C and all samples S of c of size m, A with input S outputs some concept c' consistent with S in time p(m);
- for all concepts c in C and all samples S of c such that $\mathcal{T}(c) \subseteq S$, \mathcal{A} with input S outputs some concept c' such that $c' \equiv c$.

Recently, formal models of teaching have been defined and studied in a lot of works [13,14,17,22,30,31,33]. For instance, Goldman and Mathias have defined a teaching model based on a pair teacher/learner:

Definition 4.2. A concept class C is *teachable* if there exists a teacher \mathcal{T} and a learner L such that for all adversary Adv

- the adversary Adv selects a target concept $c \in C$ and gives it to \mathcal{T} ;
- the teacher \mathcal{T} computes a teaching set $\mathcal{T}(c)$ and gives it to Adv;
- the adversary adds properly labeled examples to $\mathcal{T}(c)$. The sample S obtained is given to the learner;
- the learner outputs some concept c' such that $c' \equiv c$.

Note that the adversary has unlimited computing power and complete knowledge of C and T. The goal of the adversary is to prevent collusion between the teacher and the learner. When the learner is polynomial in the size of its input, the class C is said to be *semi-poly teachable*. When both the learner and the teacher are polynomial, the class C is said to be *poly teachable*.

The following result has been proved in [17]:

Claim 4.3. A concept class C is semi-poly teachable if and only if it is learnable from given data.

Angluin has defined in [2] a learning model from example-based queries (such as equivalence queries, membership queries and others). Next result shows the connection between these two last models.

Theorem 4.4. [14] If a concept class C is exactly learnable in deterministic polynomial-time using example-based queries then it is semi-poly teachable.

Monotone DNF formulae, decision trees are exactly learnable from membership and equivalence queries [2, 6]; consequently these concept classes are semi-poly teachable. In order to compare the PAC model under helpful distributions and the teaching model of Goldman and Mathias, we prove the following result:

Theorem 4.5. If a concept class C is semi-poly teachable, then it is PAC learnable under helpful distributions.

Proof. Let us consider a concept class C which is teachable with learner L and teacher T. We consider the same teacher T = T in the helpful learning model. A PAC learning algorithm \mathcal{A} for C under helpful distributions iterates over larger guesses for $1/D_{\min}(c)$. At each guess a sample S is drawn: enough examples are drawn so that if the guess is correct, then we have: $Pr(\mathcal{T}(c) \subseteq S) \geq 1 - \delta$. At each guess, a hypothesis testing algorithm is applied to L(S) as halting criterion. The reader should note that, due to the halting condition, learning is not probably exact.

As an immediate consequence of Theorems 4.4 and 4.5, we obtain:

Corollary 4.6. If a concept class C is exactly learnable in deterministic polynomial-time using example-based queries, then it is PAC learnable under helpful distributions.

The converse of this result is false because singletons are PAC learnable but are not learnable in several models of example-based queries [2].

Monotone DNF formulae and decision trees are exactly learnable in deterministic polynomial-time using example-based queries. Due to Corollary 4.6, they are also PAC learnable under helpful distributions. We have proved in Section 3 that DNF formulae and decision lists are PAC learnable under helpful

distributions. These classes are not known to be semi-poly teachable. These results are summarized in the following table:

class	queries	teaching model	helpful distributions
monotone DNF	yes (Ang 88)	yes (GM 96)	yes (Cor. 4.6)
decision trees	yes (Bsh 95)	yes (GM 96)	yes (Cor. 4.6)
singletons	no (Ang 88)	yes	yes
DNF	?	?	yes (Sect. 3)
decision lists	?	?	yes (Sect. 3)

5. SIMPLE PAC LEARNING MYODELS

5.1. Kolmogorov complexity and universal distribution

The reader may refer to [21] for complete definitions, proofs and guide lines in the literature. We consider the prefix variant K of Kolmogorov complexity. It is defined for $x \in \{0,1\}^*$ as follows. We consider prefix Turing machines, *i.e.* machine for which the set of programs that make them halt is a prefix code. It is also convenient to allow a Turing machine T to have some extra information y. We write T(p,y) = x to mean that Turing machine T with input p and y terminates with output x. Let us define $K_T(x|y) = \min\{|p| \mid T(p,y) = x\}$ or ∞ if such p does not exist, and $K_T(x) = K_T(x|\epsilon)$. Prefix Turing machines can be effectively enumerated and there exists a universal prefix Turing machine U. The Invariance theorem states that, for any prefix Turing machine T, there is a constant c_T such that for all strings $x, y, K_U(x|y) \leq K_T(x|y) + c_T$. Moreover, for each pair of universal prefix Turing machines U and U', there is a constant $c_{U,U'}$, such that for all strings $x, y, |K_U(x|y) - K_{U'}(x|y)| \leq c_{U,U'}$. Therefore, the Kolmogorov complexity of a string s provides an absolute measure of the information content of s, up to an additive constant. We fix a universal machine, say U, and call it the reference prefix Turing machine, and define $K(x) = K_U(x)$, and $K(x|y) = K_U(x|y)$. It can be proved that $\sum_{x \in \Sigma^*} 2^{-K(x|y)} < 1$. Let $r \in \Sigma^*$, we define $\mathbf{m}_r(x) = \lambda_r 2^{-K_U(x|r)}$ where λ_r is a normalization parameter such that $\sum_{x\in\Sigma^*}\mathbf{m}_r(x)=1$. Let $\mathbf{m}(x)=\mathbf{m}_\epsilon(x)$. The function \mathbf{m} is called the Solomonoff-Levin distribution.

5.2. SIMPLE PAC LEARNING MODELS

Definition 5.1. [20] The definition of a *simple PAC learning algorithm* is the same as the definition of a PAC learning algorithm except that the class of probability distributions is restricted to the universal Solomonoff-Levin distribution **m**.

Note that any PAC learnable class is simple PAC learnable.

Some classes, which we do not know whether they are PAC learnable, have been shown simple PAC learnable in [20]. For instance, consider the class $\log n$ -DNF of DNF formulas over n variables, where each monomial term is of Kolmogorov complexity $O(\log n)$. This is a superset of the class of k-DNF formulas and this

class is simple PAC learnable. Castro and Balcàzar have proved that $\log n$ decision lists are simple PAC learnable in [7].

In [10], we have defined another simple PAC learning model. The underlying idea is to suppose that the teacher has a representation of the target concept in mind and draws simple examples relatively to this representation.

Definition 5.2. The definition of a PAC learning algorithm with simple examples is the same as the definition of a PAC learning algorithm except that the class of probability distributions is restricted to the universal Solomonoff–Levin distribution \mathbf{m}_c , where c is the target concept.

In this simple PAC learning model, the examples are drawn according to \mathbf{m}_c instead of \mathbf{m} . The reader should note that the model is not a fixed distribution model. In the simple PAC learning model, not only classes of simple concepts are proved learnable. For instance, it can be shown that the class of DNF formulas is learnable in this model.

We will say *simple PAC learnable* for the simple PAC learning model of Li and Vitányi, *PACS learnable* for the simple PAC learning model of Denis *et al.* [8] have studied connections between the exact query model and these two models. We define in the next Section a new simple PAC Learning model.

5.3. PAC LEARNING WITH SIMPLE TEACHER

Definition 5.3. Let C be a concept class. \mathcal{T} is a *simple teacher* if there exists a constant k > 0 satisfying:

$$\forall c \in C \ \forall (x, c(x)) \in \mathcal{T}(c) \ K(x|c) \le k \log(|c|).$$

Recall that K(x|c) is the conditional complexity of x w.r.t. c, *i.e.* the length of a least self delimited program which computes x from c. We note that the simplicity of a teacher does not depend on the reference prefix Turing machine U. Note also that the polynomial size condition on teaching sets is satisfied because the cardinality of the set of strings of length lower than $k \log(|c|)$ is bounded by $|c|^{k+1}$.

Proposition 5.4. Let C be a concept class. Any computable teacher \mathcal{T} is a simple teacher.

Proof. Let c be a concept in C. Let (x,c(x)) be an example in $\mathcal{T}(c)$. We have: $K(x|c) \leq K(x|\mathcal{T}(c)) + K(\mathcal{T}(c)|c) + O(1)$. Since $Card(\mathcal{T}(c)) \leq |c|^k$, we get $K(x|\mathcal{T}(c)) \leq k \log(|c|) + O(1)$. Moreover, since \mathcal{T} is computable, $K(\mathcal{T}(c)|c) \leq O(1)$.

For instance, the teacher we have defined for the class of decision lists is computable; therefore it is a simple teacher. We now define the PAC learning model with simple teacher:

Definition 5.5. Let C be a concept class. C is PAC learnable with simple teacher if C is PAC learnable under helpful distributions for some simple teacher.

Note that the definition of PAC learnability with simple teacher does not depend on the reference prefix Turing machine U. As a corollary of Proposition 5.4, we get:

Corollary 5.6. Let C be a concept class. If C is PAC learnable under helpful distributions for a computable teacher, then C is PAC learnable with simple teacher.

Therefore decision lists are PAC learnable with simple teacher.

5.4. Comparison

We show below that learnability with simple teacher implies PACS learnability.

Proposition 5.7. Let C be a concept class. If C is PAC learnable with simple teacher then C is PACS learnable.

Proof. Let \mathcal{T} be a simple teacher such that C is PAC learnable with \mathcal{T} . Thus, there exists a constant k such that

$$\forall c \in C \ \forall (x, c(x)) \in \mathcal{T}(c) \ K(x|c) \le k \log(|c|).$$

Now, by definition of \mathbf{m}_c , we get

$$\forall c \in C \ \forall (x, c(x)) \in \mathcal{T}(c) \ \mathbf{m}_c(x) = \lambda_c 2^{-K(x|c)} \ge \lambda_c (|c|)^{-k}$$

therefore,

$$\mathbf{m}_{c_{\min}}(c) \ge \lambda_c(|c|)^{-k}$$

where $\mathbf{m}_{c_{\min}}(c)$ is defined in equation (1). Therefore \mathbf{m}_c is (really) helpful for c, \mathcal{T} . Consider now the PAC learning algorithm \mathcal{A} for C under helpful distributions; \mathcal{A} runs in polynomial time in $1/\epsilon$, $1/\delta$, l, and $1/\mathbf{m}_{c_{\min}}(c)$ where $l \geq |c|$. Consequently, \mathcal{A} runs in polynomial time in $1/\epsilon$, $1/\delta$, and l.

Consequently, the classes of decision lists and DNF formulas are PACS learnable.

In order to prove that simple classes of concepts are simple PAC learnable in the sense of [20], we can often show that a more general class is learnable with simple teacher and then use the next proposition.

Proposition 5.8. Let C be a concept class. Suppose C is PAC learnable with a simple teacher \mathcal{T} . Let k be an integer and let us define the concept class

$$C_k = \{c \in C \mid \forall (x, c(x)) \in \mathcal{T}(c) \mid K(x) \le k \log(|c|)\}$$

For each k, the concept class C_k is simple PAC learnable.

The proof is omitted because it is similar to the proof of Proposition 5.7. Decision lists are PAC learnable with simple teacher (Sect. 3). Using Proposition 5.8, we obtain a new proof of the the following result:

Proposition 5.9. [7] The class of $\log n$ -decision lists is simple PAC learnable.

The reader should note that our proof is based on an Occam algorithm. In their proof [7] directly design a simple learning algorithm which is derived from a learning algorithm from equivalence queries. Also, we give another proof of the following result:

Theorem 5.10. [8] If a concept class C is exactly learnable in deterministic polynomial-time using example-based queries then it is PACS learnable.

Proof. First, using Corollary 4.6, concept classes using example-based queries are PAC learnable under helpful distributions. It is easy to see in the proof of Theorem 4.4 by [14] that the teacher is computable and that the teaching set has polynomial size. The reader should note that the teacher could be computationally unbounded. Now, from Proposition 5.4, the teacher is also simple. We obtain the result using Proposition 5.7.

Reference [8] also proved that PACS learnability results could be derived from simple PAC learnability results if the learning algorithm is independent of the reference prefix turing machine.

6. Conclusion

The PAC learning model under helpful distributions gives a proper account of what is intended by PAC learning from characteristic examples. It extends previously defined teaching models to the PAC setting. This model is to the standard PAC model of Valiant roughly as the teaching model of Goldman and Mathias is to the identification in the limit model of Gold. The Occam's razor theorem and its converse contribute to give to our model a theoretical validation. The notion of Occam algorithm that we use, for which the length of the output hypothesis depends on the frequencies of the useful examples in the learning sample, seems of particular interest. Boolean classes, as large as the class of decision lists, are learnable in this model whereas they seem to be unreachable in the standard model. Moreover, our model allows to define a new simple PAC learning model avoiding some drawbacks of the previously defined simple PAC learning models like non computability of the Solomonoff–Levin distribution and independence w.r.t. the reference prefix Turing machine.

However, our model is maybe too weak. It may be seen as providing a necessary condition of learnability (as the standard PAC model would provide a sufficient condition of learnability).

A general difficulty when designing teaching models is to prevent collusion, *i.e.* the possibility both for the teacher to encode the target concept within the useful examples and for the learner to use this encoding. Note that there does not exist yet any more precise definition of collusion. Therefore, it is impossible to verify formally if our model prevent collusion. The only thing we can do is to check whether some class can be learned using a trivial algorithm or whether a given learning algorithm uses directly some encoding information of the target concept. It is easy to verify that representation classes of boolean functions such as DNF

or decision lists are numerous enough to prevent the encoding of target concepts by a bounded number of examples. Our learning algorithm of decision lists uses a polynomial set of examples from which it is possible to construct a superset of the monomials that compose the target concept. Is this the beginning of an encoding of the target concept? Can a helpful example bear no information on the target concept? We think that collusion in learning framework remains a challenging problem.

We think that most of problems of collusion occur when trying to learn languages from useful examples, as for languages the length of an example is not bounded. In the case of variable-length examples, any computable encoding of the target is a (very) useful example. And, for all teaching models, it is possible to render trivial any learning algorithm. For instance, DFA are learnable from membership and equivalence queries [1], therefore DFA are semi-poly teachable. A teaching set can be defined using the algorithm RPNI (see [25]). Adding the encoding of the target automaton to this teaching set, a learning algorithm may be designed as follows: decode each example, make a consistency test. The case of variable-length examples is also problematic for simple PAC models since arbitrarily long examples are simple and a learning algorithm can use long examples to increase its computational time. A discussion on collusion and on simple PAC learning model can be found in [9, 26–28].

APPENDIX A. PROOF OF THE OCCAM'S RAZOR THEOREM

Let C be a concept class and let \mathcal{T} be a polynomial teacher for C. Let k be a constant such that for every concept c, $Card(\mathcal{T}(c)) \leq (|c|)^k$. Let B be an Occam algorithm for C, \mathcal{T} with constants (a, b, α) . Let q denote the polynomial such that B has time complexity q(|c|, Card(S)). Let c be a concept. Whenever we consider the oracle EX(c, D), we suppose that the probability distribution D is helpful w.r.t. c and \mathcal{T} . Let l be an integer such that $|c| \leq l$. We first prove some technical lemmas.

The first lemma states that, for a large enough sample, we can estimate the parameter $D_{\min}(c)$.

Lemma A.1. Let p be an integer such that $p \geq 1/D_{\min}(c)$. Make $N_1(\delta, l, p) = \lceil 8p \log(l^k/\delta) \rceil$ calls to EX(c, D). This defines a sample S. Then the probability that $f_{\min}(S, c) \geq D_{\min}(c)/2$ is at least $1 - \delta$.

Proof. The proof is based on the classical Chernoff bounds (see for instance [19], pp. 190-192). \Box

The following classical lemma shows that an Occam algorithm applied to a large enough sample outputs a good approximation of the target with high probability.

Lemma A.2. Make $N_2(\epsilon, \delta, l, p) = \lceil ((\log(1/\delta) + a(2pl)^b + 1)/\epsilon)^{\frac{1}{1-\alpha}} \rceil$ calls to EX(c, D). This defines a sample S. Suppose that $f_{\min}(S, c) \geq 1/(2p)$. Let c' be the output of the Occam algorithm B on input S. Then the probability that $error(c') > \epsilon$ is at most δ .

Proof. This proof is based on the original proof in [4]. Let S be a sample of cardinality N and let us consider the set

$$H_c = \{c' \in C \mid error(c') > \epsilon \text{ and } |c'| \le a(2pl)^b N^{\alpha} \}.$$

Let $c' \in H_c$. The probability that a call to EX(c,D) returns an example consistent with c' is less than $1-\epsilon$. Therefore the probability that N calls to EX(c,D) return a sample S consistent with c' is less than $(1-\epsilon)^N$. Moreover, the cardinality of H_c is less than $2^{a(2pl)^bN^\alpha+1}$. Consequently the probability that the sample S of cardinality N is consistent with a concept c' in H_c is less than $2^{a(2pl)^bN^\alpha+1}(1-\epsilon)^N$. Verify that $2^{aN^\alpha(2pl)^b+1}(1-\epsilon)^N \leq \delta$ whenever $N \geq N_2(\epsilon, \delta, l, p)$.

Now let us suppose that $N \geq N_2(\epsilon, \delta, l, p)$ and $f_{\min}(S, c) \geq 1/(2p)$. $f_{\min}(S, c) > 0$, thus $\mathcal{T}(c) \subseteq S$. On input S the algorithm B outputs a hypothesis concept c' consistent with S such that

$$|c'| \le a(|c|/f_{\min}(S,c))^b (Card(S))^{\alpha} \le a(2pl)^b N^{\alpha}.$$

Finally if $error(c') > \epsilon, c' \in H_c$.

In the usual PAC setting, there exists an algorithm TEST which tests whether an hypothesis is a good approximation of the target concept. The hypothesis testing algorithm TEST with parameters ϵ , δ , i, and c' [16] makes $\lceil (32/\epsilon)(i \ln 2 + \ln 2/\delta) \rceil$ calls to EX(c,D) to test hypothesis c'. It accepts the hypothesis if the hypothesis is wrong on no more than a fraction of $\frac{3}{4}\epsilon$ of the examples returned by the oracle, and rejects it otherwise. $TEST(\epsilon,\delta,i,c')$ is polynomial in $1/\epsilon,1/\delta,i,|c'|$. And we have:

Lemma A.3. [16] The test $TEST(\epsilon, \delta, i, c')$ has the property that: when $error(c') \geq \epsilon$, the probability is at most $\delta/2^{i+1}$ that the test will accept c', when $error(c') \leq \epsilon/2$, the probability is at most $\delta/2^{i+1}$ that the test will reject c'.

We now give the PAC learning algorithm under helpful distributions:

```
PAC Learning Algorithm \mathcal{A} for C, \mathcal{T} input: \epsilon, \delta, l begin  \begin{array}{l} \text{Set } S \text{ to } \emptyset \text{ --} S \text{ is the current sample} \\ \text{Set } p \text{ to } 1 \text{ --} p \text{ is the current guess for } 1/D_{\min}(c) \\ \textbf{loop} \\ \text{Set } N \text{ to } \sup\{N_1(\delta/3,l,p),N_2(\epsilon/2,\delta/3,l,p)\} \\ \text{Call } EX(c,D) \text{ until } Card(S) = N \\ \text{Run at most } q(l,N) \text{ steps of } B \text{ on input } S \\ \textbf{if } B \text{ outputs some } c' \text{ and } c' \text{ consistent with } S \\ \textbf{and } |c'| \leq a(pl)^b N^\alpha \text{ and } TEST(\epsilon,\delta/3,p,c') \\ \textbf{then output } c' \text{ and halt } \\ \textbf{endif} \end{array}
```

Set p to p+1 endloop end

If \mathcal{A} halts at some step p, the hypothesis testing condition TEST is satisfied. When $error(c') > \epsilon$, the probability is at most $\delta/(3 \times 2^{p+1})$ that the test accepts c' (Lem. A.3). If, at some step p, B outputs some wrong hypothesis c', the probability that \mathcal{A} halts is less than the probability that TEST is satisfied, and therefore is less than $\delta/(3 \times 2^{p+1})$. The probability that \mathcal{A} halts with a wrong hypothesis is less than the probability that there exists some p such that B outputs some wrong hypothesis c' satisfying TEST. Consequently, the probability that \mathcal{A} halts with a wrong hypothesis is less than $\delta/6$.

It remains to prove that, with probability at least $1-\delta$, \mathcal{A} halts in polynomial time in $1/\epsilon$, $1/\delta$, l, and $1/D_{\min}(c)$. Let $p=\lceil 1/D_{\min}(c)\rceil$ and $N\geq N_1(\delta/3,l,p)$. Then, the probability that $f_{\min}(S,c)\geq D_{\min}(c)/2$ is at least $1-\delta/3$ (Lem. A.1). Suppose $f_{\min}(S,c)\geq D_{\min}(c)/2$ and $N\geq N_2(\epsilon/2,\delta/3,l,p)$. Then, the probability that $error(c')>\epsilon/2$ is at most $\delta/3$ (Lem. A.2); the probability that $error(c')\leq\epsilon/2$ is at least $1-\delta/3$ (Lem. A.2). Let us suppose that B outputs some c' with $error(c')\leq\epsilon/2$, the probability that TEST will reject c' is at most $\delta/(3\times 2^{p+1})$ (Lemma A.3). Therefore, the probability that the algorithm \mathcal{A} does not halt at step $p=\lceil 1/D_{\min}(c)\rceil$ is at most δ and the probability that the learning algorithm \mathcal{A} does not halt before $p=\lceil 1/D_{\min}(c)\rceil$ is at most δ . It is now easy to verify that if the algorithm \mathcal{A} halts before $p=\lceil 1/D_{\min}(c)\rceil$, then the running time is bounded by a polynomial in $1/\epsilon$, $1/\delta$, l, and $1/D_{\min}(c)$. This ends the proof.

APPENDIX B. PROOF OF THE CONVERSE OF THE OCCAM'S RAZOR THEOREM

In the PAC framework, the proof may be found in [5]. Let S be a sample of c such that $\mathcal{T}(c) \subseteq S$. Let D be the uniform distribution on S. Recall that $D_{\min}(c) = f_{\min}(S, c)$. Let A be the algorithm which computes the closure under exceptions in concept class C. Let A be a usually polynomial time PAC learning algorithm under helpful distributions which only takes as input ϵ and δ .

There exists k such that \mathcal{A} halts in time $(\frac{|c|}{\epsilon\delta D_{\min}(c)})^k$ with probability at least $1-\delta$. We now give the randomized algorithm \mathcal{B} :

Randomized Occam Algorithm \mathcal{B} for C, \mathcal{T} input: δ , S begin $\operatorname{Set} \epsilon \text{ to } ||S||^{-\frac{1}{k+1}}$

Dovetail $\lceil \log(1/\delta) \rceil$ executions of $\mathcal{A}(\epsilon, 1/4)$ Pick the first h such that $|\{x \mid (x, c(x)) \in S \ c(x) \neq h(x)\}| \leq \epsilon Card(S)$ Output $A(h, \{x \mid (x, c(x)) \in S \ c(x) \neq h(x)\})$ and halt **end**

When \mathcal{A} takes as input ϵ and 1/4, \mathcal{A} halts in time lower than $t = (\frac{4|c|}{\epsilon D_{\min}(c)})^k$ with probability at least 3/4 because \mathcal{A} is a usually polynomial time PAC learning algorithm under helpful distributions. D is the uniform distribution on S, thus when \mathcal{A} takes as input ϵ and 1/4, \mathcal{A} outputs a concept h such that

$$\sharp exceptions(h) = |\{x \mid (x, c(x)) \in S \ c(x) \neq h(x)\}| \leq \epsilon Card(S)$$

with probability at least 3/4. One execution of $\mathcal{A}(\epsilon, 1/4)$ halts in time lower than t and outputs a concept h such that $\sharp exceptions(h) \leq \epsilon Card(S)$ with probability at least 1/2. The probability such that none of the $\lceil \log(1/\delta) \rceil$ executions of $\mathcal{A}(\epsilon, 1/4)$ halts in time lower than t and outputs a concept h such that $\sharp exceptions(h) \leq \epsilon Card(S)$ is lower than $(1/2)^{\lceil \log(1/\delta) \rceil} \leq \delta$. Consequently, the algorithm outputs some concept h such that $\sharp exceptions(h) \leq \epsilon Card(S)$ with probability at least $1 - \delta$.

Now incorporate the exceptions to concept h, *i.e.* compute $c' = A(h, \{x \mid (x, c(x)) \in S \ c(x) \neq h(x)\})$. Using the strong closure under exceptions property, we get:

$$|c'| \le \alpha(|h| + \epsilon||S||)\log(|h| + \epsilon||S||) + \beta.$$

There exists a such that

$$|c'| \le a(|h| + \epsilon||S||)^{(1 + \frac{1}{2k})}$$
.

The size of h is bounded by the computation time of \mathcal{A} and $\epsilon = (||S||)^{-\frac{1}{k+1}}$, therefore

$$|c'| \leq a \left(\left(\frac{4|c|}{\epsilon D_{\min}(c)} \right)^k + \left(||S||^{\frac{k}{k+1}} \right)^{\left(1 + \frac{1}{2k}\right)}$$

$$|c'| \leq a (||S||)^{\left(\frac{k}{k+1}\right) \times \left(1 + \frac{1}{2k}\right)} \left(1 + \left(\frac{4|c|}{\epsilon D_{\min}(c)} \right)^k (||S||)^{\frac{-k}{k+1}} \right)^{\left(1 + \frac{1}{2k}\right)}$$

$$|c'| \leq a (||S||)^{\frac{k}{k+1} \times \left(1 + \frac{1}{2k}\right)} \left(1 + \left(\frac{4|c|}{D_{\min}(c)} \right)^k \right)^{\left(1 + \frac{1}{2k}\right)} .$$

Lastly, let us note that $\alpha = (k/k+1) \times (1+\frac{1}{2k}) = \frac{2k+1}{2k+2} < 1$. Therefore,

$$|c'| \le a(Card(S))^{\alpha} n^{\alpha} \left(1 + \left(\frac{4|c|}{D_{\min}(c)}\right)^k\right)^{(1+\frac{1}{2k})}.$$

Thus, as $|c| \geq n$, we obtain the expected bound on |c'|. Since the run of our algorithm is clearly polynomial in the various parameters, we have shown that \mathcal{B} is a randomized Occam algorithm for C, \mathcal{T} .

Acknowldegements. The authors thank an anonymous referee for helpful comments on this paper.

References

- D. Angluin, Learning Regular Sets from Queries and Counterexamples. Inform. and Comput. 75 (1987) 87-106.
- [2] D. Angluin, Queries and Concept Learning. Machine Learning 2 (1988) 319-342.
- [3] G.M. Benedek and A. Itai, Nonuniform Learnability, in ICALP (1988) 82-92.
- [4] A. Blumer, A. Ehrenfeucht, D. Haussler and M.K. Warmuth, Occam's Razor. Inform. Process. Lett. 24 (1987) 377-380.
- [5] R. Board and L. Pitt, On the Necessity of Occam Algorithms. Theoret. Comput. Sci. 100 (1992) 157-184.
- [6] N.H. Bshouty, Exact Learning Boolean Function via the Monotone Theory. Inform. and Comput. 123 (1995) 146-153.
- [7] J. Castro and J.L. Balcázar, Simple PAC learning of simple decision lists, in ALT 95, 6th International Workshop on Algorithmic Learning Theory. Springer, Lecture Notes in Comput. Sci. 997 (1995) 239-250.
- [8] J. Castro and D. Guijarro, PACS, simple-PAC and query learning. Inform. Process. Lett. 73 (2000) 11-16.
- [9] F. Denis, Learning regular languages from simple positive examples, Machine Learning.
 Technical Report LIFL 321 1998; http://www.lifl.fr/denis (to appear).
- [10] F. Denis, C. D'Halluin and R. Gilleron, PAC Learning with Simple Examples, in 13th Annual Symposium on Theoretical Aspects of Computer Science. Springer-Verlag, Lecture Notes in Comput. Sci. 1046 (1996) 231-242.
- [11] F. Denis and R. Gilleron, PAC learning under helpful distributions, in Proc. of the 8th International Workshop on Algorithmic Learning Theory (ALT-97), edited by M. Li and A. Maruoka. Springer-Verlag, Berlin, Lecture Notes in Comput. Sci. 1316 (1997) 132-145.
- [12] E.M. Gold, Complexity of Automaton Identification from Given Data. Inform. and Control 37 (1978) 302-320.
- [13] S.A. Goldman and M.J. Kearns, On the Complexity of Teaching. J. Comput. System Sci. 50 (1995) 20-31.
- [14] S.A. Goldman and H.D. Mathias, Teaching a Smarter Learner. J. Comput. System Sci. 52 (1996) 255-267.
- [15] T. Hancock, T. Jiang, M. Li and J. Tromp, Lower Bounds on Learning Decision Lists and Trees. Inform. and Comput. 126 (1996) 114-122.
- [16] D. Haussler, M. Kearns, N. Littlestone and M.K. Warmuth, Equivalence of Models for Polynomial Learnability. *Inform. and Comput.* 95 (1991) 129-161.
- [17] C.D.L. Higuera, Characteristic Sets for Polynomial Grammatical Inference. *Machine Learning* 27 (1997) 125-137.
- [18] M. Kearns, M. Li, L. Pitt and L.G. Valiant, Recent Results on Boolean Concept Learning, in Proc. of the Fourth International Workshop on Machine Learning (1987) 337-352.
- [19] M.J. Kearns and U.V. Vazirani, An Introduction to Computational Learning Theory. MIT Press (1994).
- [20] M. Li and P.M.B. Vitányi, Learning simple concepts under simple distributions. SIAM J. Comput. 20 (1991) 911-935.
- [21] M. Li and P. Vitányi, An introduction to Kolmogorov complexity and its applications, 2nd Edition. Springer-Verlag (1997).

- [22] D.H. Mathias, DNF: If You Can't Learn 'em, Teach 'em: An Interactive Model of Teaching, in Proc. of the 8th Annual Conference on Computational Learning Theory (COLT'95). ACM Press, New York (1995) 222-229.
- [23] B.K. Natarajan, Machine Learning: A Theoretical Approach. Morgan Kaufmann, San Mateo, CA (1991).
- [24] B.K. Natarajan, On Learning Boolean Functions, in Proc. of the 19th Annual ACM Symposium on Theory of Computing. ACM Press (1987) 296-304.
- [25] J. Oncina and P. Garcia, Inferring regular languages in polynomial update time, in Pattern Recognition and Image Analysis (1992) 49-61.
- [26] R. Parekh and V. Honavar, On the Relationships between Models of Learning in Helpful Environments, in Proc. Fifth International Conference on Grammatical Inference (2000).
- [27] R. Parekh and V. Honavar, Learning DFA from simple examples, in Proc. of the 8th International Workshop on Algorithmic Learning Theory (ALT-97), edited by M. Li and A. Maruoka. Springer, Berlin, Lecture Notes in Artificial Intelligence 1316 (1997) 116-131.
- [28] R. Parekh and V. Honavar, Simple DFA are polynomially probably exactly learnable from simple examples, in Proc. 16th International Conf. on Machine Learning (1999) 298-306.
- [29] R.L. Rivest, Learning Decision Lists. Machine Learning 2 (1987) 229-246.
- [30] K. Romanik, Approximate Testing and Learnability, in Proc. of the 5th Annual ACM Workshop on Computational Learning Theory, edited by D. Haussler. ACM Press, Pittsburgh, PA (1992) 327-332.
- [31] S. Salzberg, A. Delcher, D. Heath and S. Kasif, Learning with a Helpful Teacher, in Proc. of the 12th International Joint Conference on Artificial Intelligence, edited by R. Myopoulos and J. Reiter. Morgan Kaufmann, Sydney, Australia (1991) 705-711.
- [32] R.E. Schapire, The Strength of Weak Learnability. Machine Learning 5 (1990) 197-227.
- [33] A. Shinohara and S. Miyano, Teachability in Computational Learning. NEWGEN: New Generation Computing 8 (1991).
- [34] L.G. Valiant, A Theory of the Learnable. Commun. ACM 27 (1984) 1134-1142.

Communicated by J.L. Balcazar.

Received February 28, 2000. Revised December 20, 2000. Accepted April 3, 2001.

To access this journal online: www.edpsciences.org