Programming Research Group

A LEARNABILITY MODEL FOR UNIVERSAL REPRESENTATIONS

Stephen Muggleton
C. David Page Jr.

PRG-TR-3-97

Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford OX1 3QD

A Learnability Model for Universal Representations

Stephen Muggleton
C. David Page Jr.

Oxford University Computing Laboratory
Wolfson Building
Parks Road
Oxford OX1 3QD
U.K.
Email: {steve,dpage}@prg.oxford.ac.uk

Abstract

This paper defines a new computational model of inductive learning, called *U-learnability* (*Universal Learnability*), that is well-suited for rich representation languages, most notably for universal (Turing equivalent) representations. It is motivated by three observations. Firstly, existing computational models of inductive learning—the best known of which are *identification in the limit* and *PAC-learnability*—either fail to provide guarantees of accuracy and computational efficiency or require severely restricted representation languages. Secondly, practical machine learning programs use rich representations yet often perform efficiently and achieve high degrees of accuracy. Thirdly, these machine learning programs often make assumptions about the underlying probability distribution over *target concepts*; such an assumption provides a *soft bias* for learning. This contrasts with existing computational models of inductive learning, which incorporate only *language biases*, or *hard biases*. U-learnability incorporates soft biases in the form of a Bayesian prior distribution over target concepts. (Other authors have argued for the use of Bayesian prior distributions in a manner more restricted than their use in U-learnability, though this has not lead to a new model of polynomial-time learnability.) In addition, due to problems of undecidability with universal representations, U-learnability makes special provision for the use of *time-bounded* concepts. A time bound takes the form of a polynomial function of the size of an example.

1 Introduction

Learning is a central feature of human cognition. As overwhelming amounts of data are becoming available in scientific, sensory and linguistic domains, algorithms which allow computers to learn are becoming increasingly important. For instance, recent papers [20, 19, 23, 36, 35] report a machine learning algorithm producing novel biological insights beyond those achieved by persistent human visual inspection of the data. The algorithm induces pure logic programs from data. Universal (Turing equivalent) representations, such as logic programs, are particularly appropriate in scientific domains due to the complexity of the objects and relationships involved. Clearly universal representations are also of interest in a number of other learning contexts, including program synthesis [37, 32]. In this paper we present a new computational model of learning which is appropriate for use with representations up to and including universal ones.

Gold [15], motivated by the problem of human language acquisition, introduced the first computational model of learning. This model, known as "identification-in-the-limit", describes the

conditions for finite convergence of a language identification procedure. Although Gold's model is well suited for learning universal representations [6, 5], it fails to meet our purposes for two reasons. Firstly, at no point can the learning agent give guarantees on the degree of error of a preferred hypothesis. Secondly, although convergence is finite it could take arbitrarily long.

Valiant [38], motivated by the needs of expert system technology, developed a model that provides guarantees on the degree of error and efficiency of convergence for learning. This model, known as Probably-Approximately-Correct (PAC) learning, has gained wide acceptance and stimulated much high-quality research (for example [8, 26, 18, 27, 2, 1]). However, PAC's stringent requirements lead to negative results for universal representations (logic programs are prediction-hard, even when severely restricted [13]). Even arbitrary propositional formulae and finite state automata are unlearnable (based on certain cryptographic assumptions) [18], and propositional formulae in disjunctive normal form are not known to be learnable. This contrasts with the fact that most machine learning algorithms use representations at least this rich.

Machine learning algorithms that use representations not known to be PAC-learnable—or known not to be PAC-learnable—make additional assumptions, outside those in the PAC model, about the distribution of problems and their examples. For instance, a very common assumption used in Machine Learning is that textually shorter theories are more likely to be correct a priori (Occam's Razor). Although Occam algorithms have been studied in PAC-learning [7], this study has not led to algorithms capable of learning in universal representations, as it has in Inductive Logic Programming [24]. The Occam assumption was made explicitly by Solomonoff [33] who proposed universality of the distribution which assigns the prior probability of hypothesis h to be $2^{-length(h)}$. More sophisticated versions of this assumption are used in Rissanen's [29, 30] Minimal Description Length (MDL) principle. A very different kind of distribution assumption was implicitly implemented by Arbab, Michie [3] and Bratko [9] in an algorithm for constructing decision trees. The algorithm constructs a decision list (linear decision tree) where one exists and otherwise returns the most linear decision tree which can be constructed from the data. One can imagine a variety of similar kinds of distributions; assigning higher prior probabilities to grammars that are regular or *almost* so, logic programs that are deterministic or *almost* so, and Turing machines that run in linear time or *almost* so. Yet another form of distribution assumption popular in "clustering" algorithms is that sets of clusters are preferred to the degree that the clusters are small, separated and dense. This works when examples are known to form small, dense groups, as in the case of celestial data [12, 11]. A very different kind of hypothesis distribution is required when learning concepts which mimic human performance in skill tasks [31]. For such tasks predictive accuracy is dependent on hypotheses being evaluable in time similar to that of human reaction. Within Statistics distributional assumptions include Fisher's [14] Maximum Likelihood Principle, which holds when all hypotheses have almost equal prior probability. According to a recent large scale set of comparative trials on industrial data [22] the assumptions in machine learning and statistical algorithms often fit real world problems; in such cases the algorithms succeed. Unlike PAC-learnability, the model in this paper allows the use of such distributional assumptions. Universal representations are shown to be polynomially learnable for certain families of distributions. Because this model facilitates the use of universal representations (as well as restricted ones) it is called *Universal-learnability*, or *U-learnability*.

Owing to non-terminating computations, the use of rich representations often requires time-bounds on hypotheses. These can take the form of limits on number of resolution steps for logic programs or bounds on the number of computation steps for a Turing machine. Within U-learnability distributions range over hypotheses which include a time bound. The use of such time bounds allows us to characterise not only inductive learning but also speed-up learning. Speed-up learn-

ing cannot be viewed as a special case of PAC-learning although Natarajan [25] has provided a PAC-style formalisation of speed-up learning, which has as yet yielded few positive results.

U-learnability has a natural interpretation within Bayesian analysis. Distributional assumptions over hypotheses equate to subjective prior distributions from which, for a given set of examples, conditional distributions can be derived using Bayes' Theorem. Bayesian approaches to Machine Learning have been discussed previously in the literature. For instance, Buntine [10] develops a Bayesian framework for learning, which he applies to the problem of learning class probability trees. Also Haussler et al. [16] analyse the sample complexity of two Bayesian algorithms. This analysis focuses on average case, rather than worst case, accuracy. Nevertheless, unlike U-learnability (Section 2), neither Buntine nor Haussler et al. considers the additional use of the prior distribution on hypotheses to measure average case, rather than worst case, time complexity. As a result, neither Buntine nor Haussler et al. develops a new learning model in which significantly more expressive representations are learnable. U-learnability differs from the work of Buntine and Haussler et al. in a number of other ways as well, including the use of a *parameterised family* of possible prior distributions on hypotheses, rather than a single prior. Furthermore, the accuracy guarantees need only hold when the sample size is greater than some particular polynomial in the parameter of the prior distribution on hypotheses.

It should be noted that the use of a distribution over hypotheses provides a vehicle for incorporating "soft" biases, or preferences, into a learning task, in addition to the standard "hard" bias imposed by a language restriction. The importance of such soft biases has long been recognised in Machine Learning (for example, see the discussion of preference criteria in [21]), but such biases have never before been incorporated into a formal computational model of learning.

The paper is arranged as follows. Section 2 defines U-learnability. Section 3 presents some initial, basic results, and Section 4 points to directions for more advanced results. Section 5 discusses extensions to the definition of U-learnability to allow the use of *background knowledge*. Section 6 gives our present conclusions about U-learnability.

2 U-learnability model (simple)

In this section we give the basic definition of U-learnability without incorporating background knowledge. In a later section this definition will be augmented.

2.1 Preliminaries for U-learning

Let $(\Sigma_E, X, \Sigma_C, R, c)$ be the representation of a learning problem, where X is the domain of examples (finite strings over the alphabet Σ_E), R is the class of concept representations (finite strings over the alphabet Σ_C), and $c : R \to 2^X$ maps concept representations to concepts (subsets of X).

Within PAC-learning it is assumed that concept representations are evaluable in time bounded by a particular polynomial function.[1] When R is Universal (Turing equivalent), problems associated with halting dictate that computations must be truncated according to a time-bound. Thus let P, a subset of the univariate polynomial functions, be a set of time-bounds. A concept representation is a pair (r, p), for $r \in R$ and $p \in P$. Let $A(r, p, x)$ be an evaluation algorithm that maps any tuple (r, p, x), for $r \in R$, $p \in P$, and $x \in X$, to 0 or 1. Let $A(r, p, x)$ run in time bounded by $p(|x|)$. Furthermore, let A have the following properties. First, for all $r \in R$ and $x \in X$, if $x \notin c(r)$ then

[1] We assume throughout the paper that any univariate polynomial function f has the form $f(x) = c_1 x^{c_2}$, and any bivariate polynomial f has the form $f(x, y) = c_1 x^{c_2} y^{c_3}$, where $c_1, c_2, c_3 > 0$.

for every $p \in P$: $A(r, p, x) = 0$. Second, for all $r \in R$ and $x \in X$, if $x \in c(r)$ then there exists $p \in P$ such that for every $p' \in P$ with $p'(|x|) \geq p(|x|)$: $A(r, p', x) = 1$.

Let F be any family of probability distributions over $R \times P$, where the distributions in F each have an associated parameter $n_F \geq 0$ (for example, n might be the greater of the mean and standard deviation of each distribution). Let G be any family of probability distributions over X, where the distributions in G each have an associated parameter $n_G \geq 0$.

The intuition behind these preliminaries is the following. First, the representation as defined simply provides the description language for examples and concepts. Second, the algorithm A provides an efficient method for determining how a given concept (together with a time bound) classifies a given example. The assymetry in the behavior of A relative to a time bound—that with more time, more examples are labeled *positive*—reflects the standard conservative use of time bounds in computing. For example, if the concept representation is the set of time-bounded definite clause theories, it is standard to say that a given goal is true according to such a program just if it is provable within the time bound; with more time, possibly more goals—but not fewer goals—become provable. Third, the families of distributions model the learner's knowledge or assumptions about true underlying distributions over hypotheses and examples. Complete knowledge, or strong assumptions, would be modelled by having particular known distributions over the hypotheses and examples. But a learner does not necessarily have such complete knowledge or such strong assumptions. Incomplete knowledge or weak assumptions are instead modelled by families of distributions that include the true distributions. Finally, some distributions within a family may make learning more *difficult* than do others. For example, a distribution in which the probabilities of hypotheses decrease rapidly as hypothesis size (or some other measure of difficulty) grows will probably make learning easier than will a distribution in which the probabilities decrease slowly with a growth in size. The parameters of the distributions in a family may provide some indication of the difficulty of the distribution.

2.2 Protocol for U-learning

In U-learning, a Teacher randomly chooses a target $(r, p) \in R \times P$, according to some distribution D_F in F. The Teacher presents to a learning algorithm L an infinite stream of labeled examples $\langle (x_1, l_1), (x_2, l_2), ... \rangle$ where: each example x_i is drawn randomly, independently of the preceding examples, from X according to some fixed, unknown distribution D_G in G and labeled by $l_i = A(r, p, x_i)$. After each example x_i in the stream of examples, L outputs a hypothesis $H_i = (r_i, p_i)$, where $r_i \in R$ and $p_i \in P$.

The intuition behind the protocol comes from a broader view of the inductive learning task than is usually taken. Consider a task such as structure-activity prediction of pharmaceutical chemicals. Structure-activity prediction is not actually a single prediction task, but a variety of tasks based on different *receptor sites*. A chemical may be active relative to one type of receptor site (that is, it may exhibit one type of activity) but not another. Therefore, the concept of "active chemical" varies with the receptor site. Hence each receptor site defines a distinct target concept, but all such concepts are closely related. Thus this machine learning problem is not defined by a *single* target concept, for which a trivial optimal algorithm exists (though we may not know what it is), but by a set of related target concepts that are drawn according to some distribution. If the language in which target concepts may be represented is complex, perhaps even universal, then we cannot expect a machine learning algorithm to perform well on *every possible* target concept. Rather, common sense tells us that a machine learning algorithm performs well on the task if, *on average* over all target concepts and all example sets, the algorithm efficiently finds an accurate concept.

(It is the job of the U-learnability definition, in the following subsection, to make this sentence precise.) We claim that many—probably most—machine learning settings (outside of artificial data sets labeled according to a single target) fit this broader view of inductive learning. For this reason the teacher begins with a target chosen randomly according to some underlying distribution within the family of possible distributions.

2.3 Definition of U-learnability

Throughout the following definition, for any positive integer m let $\vec{X}_m$ denote $\langle x_1, ..., x_m \rangle$, a vector of m examples from X.

Definition 1 U-learnability. *Let F and G be families of distributions over $R \times P$ and X respectively. The pair (F,G) is* U-learnable *just if there exist a learning algorithm L and three polynomial functions,* LEARNTIME-BOUND(x), DELAY-BOUND(x, y), *and* ERROR-BOUND(x), *such that for every distribution D_F (with parameter n_F) in F, and every distribution D_G (with parameter n_G) in G, the following hold.*

Time Complexity. *The average-case time complexity of L at any point in a run is bounded by* LEARNTIME-BOUND(M), *where M is the sum of $p(|x_i|)$ over the examples x_i seen to that point.*[2]

Correctness. *For all $m \geq$* DELAY-BOUND(n_F, n_G)*:*

$$\sum_{all\ (r,p,\vec{X}_{m+1})} [Pr_{D_F,D_G}(r, p, \vec{X}_{m+1})][E_L(r, p, \vec{X}_{m+1})] < \text{ERROR-BOUND}(\frac{1}{m})$$

where $E_L(r, p, \vec{X}_{m+1}) = 0$ if L correctly classifies example x_{m+1} given examples $\langle x_1, ..., x_m \rangle$ labeled according to (r, p), and $E_L(r, p, \vec{X}_{m+1}) = 1$ otherwise.

As with PAC-learning, we can also consider a *prediction* variant, in which the Teacher draws a target from $R \times P$ but the learner draws hypotheses from any class $R' \times P'$ with a possibly different associated evaluation algorithm A'. Also as with PAC-learning, we can consider variants in which the learner sees positive examples only (or negative examples only). In other words, the learner receives only the positive examples (or only the negative examples) that are drawn randomly, independently from X according to D_G. Note that in this case, the definition of correctness cannot be based on the performance of the hypothesis H_m on the example x_{m+1}, since x_{m+1} is necessarily a positive example (or necessarily a negative example). Instead, the definition must be based on the performance of H_m on some independent example, which may be positive or negative, drawn randomly according to D_G.

[2] To be more precise, let m be any positive integer. Let $Pr_{D_F}(r, p)$ be the probability assigned to (r, p) by D_F, and (with a slight abuse of notation) let $Pr_{D_F,D_G}(r, p, \vec{X}_m)$ be the product of $Pr_{D_F}(r, p)$ and the probability of obtaining the sequence $\vec{X}_m$ when drawing m examples randomly and independently according to D_G. Let $\text{TIME}_L(r, p, \vec{X}_m)$ be the time spent by L until output of its hypothesis H_m, when the target is (r, p) and the initial sequence of m examples is $\vec{X}_m$. Then we say that the average-case time complexity of L to output H_m is bounded by LEARNTIME-BOUND$(x) = x^c$ (with some appropriate coefficient) if the sum (or integral), over all tuples $(r, p, \vec{X}_m)$, of

$$[Pr_{D_F,D_G}(r, p, \vec{X}_m)]\frac{(\text{TIME}_L(r, p, \vec{X}_m))^{\frac{1}{c}}}{M_{(r,p,\vec{X}_m)}}$$

is less than infinity, where $M_{(r,p,\vec{X}_m)}$ is the sum of $p(|x_i|)$ over all x_i in $\vec{X}_m$. See [4] for the motivation of this definition of average-case complexity.

Notice that the definition of U-learnability is superficially similar to identification in the limit [15]. Nevertheless neither identification-in-the-limit nor U-learnability is a special case of the other, because examples are drawn in very different ways in the two models. The following remark notes that PAC-learnability *is* a special case of U-learnability.

Remark 2 U-learnability generalises PAC. *Let $(\Sigma_E, X, \Sigma_C, R, c)$ be any representation such that for some evaluation algorithm A and bivariate polynomial p: A classifies any $x \in X$ according to any $r \in R$ in time bounded by $p(|r|, |x|)$. (The existence of A and p is a standard requirement in PAC-learning.) For any $r \in R$, let p_r denote the univariate polynomial such that: for all y, $p_r(y) = p(|r|, y)$. Let F be a family of distributions that contains one distribution D_i for each $r_i \in R$, such that D_i assigns probability 1 to (r_i, p_{r_i}). Let $n = 0$ be the parameter for each member of F. Let G be the family of all distributions over X. Then the representation $(\Sigma_E, X, \Sigma_C, R, c)$ is PAC-learnable (PAC-predictable) if and only if (F, G) is U-learnable (U-predictable).*

3 Results for simple U-learnability

This section presents initial results about simple U-learnability. In each of these results, let $(\Sigma_E, X, \Sigma_C, R, c)$ be the representation of any given learning problem.

Before presenting these results, we develop a useful property similar to the *Blumer Bound* [8] that is used in many PAC-learning results. (The Blumer Bound states that for a finite hypothesis space H, a sample of size $m \geq \frac{\ln|H| + \ln\frac{1}{\delta}}{\epsilon}$ is sufficient for accuracy $1 - \epsilon$ with confidence $1 - \delta$.) Let $H = \langle H_1, H_2, ..., H_k \rangle$ be a sequence of sets of hypotheses, and let D_F be a distribution over the hypotheses in H. For all i, let $Pr_{D_F}(H_i)$ denote the sum of the probabilities (under D_F) of the hypotheses in H_i that do not appear in some H_j for $j < i$. Furthermore, let $[H_i]$ denote the subsequence $\langle H_1, ..., H_i \rangle$ of H. Let $|H_i|$ denote the number of hypotheses in H_i, and let $|[H_i]|$ denote $|H_1| + ... + |H_i|$. Then we have the following useful result.

Theorem 3 (Average-case Blumer Bound) *Let S be any sample of m examples drawn randomly and independently from a domain X according to a distribution D_G on X, and labeled according to a target concept (r, p) that is drawn randomly according to a distribution D_F on H. Let h be any hypothesis in H such that:*

- *h is consistent with S*
- *$h \in H_i$ where: for all $1 \leq j < i$, H_j does not contain a hypothesis that is consistent with S.*

We say that h is an earliest consistent hypothesis. *Let $ES(H)$ denote*

$$\sum_{i=1}^{k} Pr_{D_F}(H_i)|[H_i]|$$

For any $0 < \epsilon, \delta < 1$, if $m \geq \frac{\ln ES + \ln\frac{1}{\delta}}{\epsilon}$ then

$$\sum_{all\ (r,p,\vec{X}_m)} [Pr_{D_F, D_G}(r, p, \vec{X}_m)][E_\epsilon(r, p, \vec{X}_m)] < \delta$$

where $E_\epsilon(r, p, \vec{X}_m) = 0$ if every earliest consistent hypothesis h is $(1-\epsilon)$-accurate, and $E_\epsilon(r, p, \vec{X}_m) = 1$ otherwise. By $(1-\epsilon)$-accurate, we mean that the probability that h misclassifies an example drawn randomly according to D_G is less than ϵ.

Proof: We may rewrite the statement

$$\sum_{\text{all } (r,p,\vec{X}_m)} [Pr_{D_F,D_G}(r,p,\vec{X}_m)][E_\epsilon(r,p,\vec{X}_m)] < \delta$$

in the theorem as

$$\sum_{\text{all } (r,p)} [Pr_{D_F}(r,p)] \sum_{\text{all } \vec{X}_m} [Pr_{D_G}(\vec{X}_m)][E_\epsilon(r,p,\vec{X}_m)] < \delta$$

Consider a given target concept (r,p). Let H_i be the first set in H containing (r,p). Then for any sample $\vec{X}_m$ labeled according to (r,p), all the earliest consistent hypotheses h must be in the same set H_j, for some $j \leq i$. Given a particular target (r,p) occurring first in H_i, the probability that *any* such h is consistent with $\vec{X}_m$ and yet is not $(1-\epsilon)$-accurate is at most $|[H_j]|(1-\epsilon)^m \leq |[H_i]|(1-\epsilon)^m$. Therefore, it is sufficient to show that

$$\sum_{i=1}^{k} [Pr_{D_F}(H_i)](|[H_i]|(1-\epsilon)^m) < \delta$$

This in turn may be rewritten as

$$(1-\epsilon)^m \sum_{i=1}^{k} [Pr_{D_F}(H_i)](|[H_i]|) < \delta$$

Substituting $ES(H)$ for

$$\sum_{i=1}^{k} Pr_{D_F}(H_i)|[H_i]|$$

yields

$$(1-\epsilon)^m (ES) < \delta$$

This holds just if

$$m \geq \frac{\ln ES + \ln \frac{1}{\delta}}{\epsilon}$$

(This last step can be verified by the arguments used to derive the Blumer Bound [7].) □

Definition 4 (Polynomial and Exponential Size-Based Enumerations) *Let R be countable. Let $E = \langle E_1, E_2, ... \rangle$ be any enumeration of non-empty subsets of R such that:*

1. *every $r \in R$ is in some E_i,*
2. *for any $r_i \in E_i$ and $r_j \in E_j$, it is the case that $|r_j| > |r_i|$ only if $j > i$, and*
3. *the members of E_i, for each i, can be determined efficiently (by an algorithm that runs in time polynomial in the sum of the sizes of the members of E_i).*

E is a polynomial size-based enumeration *of R if there exists a polynomial p such that: for all $i \geq 1$, the cardinality of E_i is at most $p(i)$. E is an* exponential size-based enumeration *of R if there exists an exponential function[3] f such that: for all $i \geq 1$, the cardinality of E_i is at most $f(i)$.*

[3] An exponential function f is taken to be a function of the form $e(x) = c^x$ for some $c > 1$.

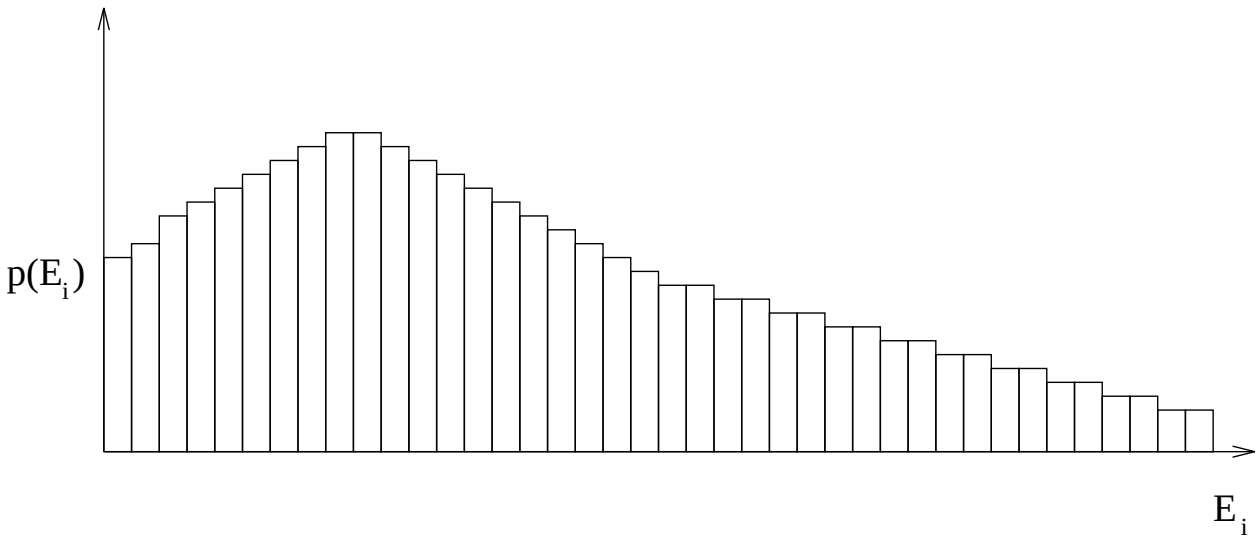

Figure 1: A distribution in D_P

Definition 5 (Polynomial Size-Based Distribution) *Let R have polynomial size-based enumeration $E = \langle E_1, E_2, ... \rangle$. Let D' be any distribution, with finite variance, over the positive integers. Let D be any distribution over R such that for each $i \geq 1$, the probabilities of the $r \in E_i$ sum to $Pr_{D'}(i)$. Then D is a* polynomial size-based distribution *over R. We specify the parameter n of D to be the maximum of the mean and standard deviation of D'.*

Definition 6 (Exponential Size-Based Distribution) *Let R have exponential size-based enumeration $E = \langle E_1, E_2, ... \rangle$, with exponential function $f(x) = c^x$. Let $D_{c'}$, for a real number $c' \geq c$, be the discrete exponential distribution with probability density function (p.d.f.)*

$$Pr(x) = (c' - 1)c'^{-x} \text{ for all integers } x \geq 1$$

Let D be any distribution over R such that for each $i \geq 1$, the probabilities of the $r \in E_i$ sum to $Pr_{D_{c'}}(i)$. Then D is an exponential size-based distribution *over R. We specify the parameter n of D to be the maximum of the mean and standard deviation of $D_{c'}$.*

Definition 7 The distribution family D_P. *Let k be a positive integer, and let P^k be the set of all univariate polynomial functions $p_{c_j}(x) = c_j x^k$ where c_j is a positive integer. For each polynomial size-based distribution D' over R, and each distribution D'' over the positive integers with the p.d.f. $Pr_{D''}(x) = x^{-c}$ for $c > 3$, we define a distribution D over all $(r_i, p_{c_j}) \in R \times P^k$ with the following p.d.f.*

$$\begin{aligned} Pr_D(r_i, p_{c_j}) &= 0 \text{ if } c_j \leq |r_i| \\ &= [Pr_{D'}(r_i)][Pr_{D''}(c_j - |r_i|)] \text{ otherwise.} \end{aligned}$$

(It is worth noting that all distributions D'' of the form specified have finite variance, and therefore finite mean and standard deviation, although the standard deviation grows arbitrarily large as c nears 3.) Let the parameter n of D be the maximum of the parameter of D' and the mean and standard deviation of D''. The distribution family D_P consists of exactly the distributions D defined in this way, for all choices of D' and D''.

The U-learning algorithm for D_P
Input: Example sequence $\langle (x_1, l_1), (x_2, l_2), \ldots \rangle$.
Output: Hypothesis sequence $\langle (r_1, p_1), (r_2, p_2), \ldots \rangle$.

1. Let $m = 1$.
2. Obtain the next labeled example (x_m, l_m) from the input sequence.
3. Let $i = 1$.
4. If for some $r \in E_i$ and $|r| < c \leq |r| + m$, the hypothesis (r, p_c) is consistent with the m examples seen thus far (that is, $A(r, p_c, x_j) = l_j$, for all $1 \leq j \leq m$), then output (r, p_c) and go to 7.
5. If $i = m$ then output (r, p_{r+1}) for $r \in E_m$ (or any other hypothesis) and go to 7.
6. Increment i and go to 4.
7. Increment m and go to 2.

Figure 2: The U-learning algorithm for D_P

Theorem 8 U-learnability under D_P. *Let the distribution family D_P be defined by a particular polynomial size-based enumeration and a particular choice of k to specify the set of polynomials P^k. Let G be any family of distributions over X. The pair (D_P, G) is U-learnable.*

Proof: The proof is constructive. It shows that the algorithm in Figure 2 U-learns (D_P, G). Time complexity is first addressed followed by correctness.

Time complexity. Let (r, p_d) be the target and let f be the least positive integer such that $r \in E_f$. We must show that there exists a univariate polynomial LEARNTIME-BOUND, such that the average case time complexity of the algorithm is bounded by LEARNTIME-BOUND(M), where

$$M = \sum_{j=1}^{m} (p_d(|x_j|))$$

For any new example (x_m, l_m), the time taken by the algorithm until output of H_m is at most the time required to test the hypotheses (r_i, p_c) for $r_i \in \langle E_1, .., E_g \rangle$ and $|r_i| < c \leq h$, where $g = min(f, m)$ and $h = min(d, |r_i| + m)$. The number of such hypotheses is at most $p'(m)m^2$, since there are $p'(m)m$ possible choices of r_i and m possible choices of c given r_i. Each hypothesis can be generated efficiently (by the definition of D_P). The time taken to test a given hypothesis (r_i, p_c) is at most

$$\sum_{j=1}^{m} (p_c(|x_j|))$$

Because $c \leq d + m$, and because $m < M$, the total time to output each H_m is bounded by a polynomial in M (LEARNTIME-BOUND).[4]

Correctness. If there exists a consistent hypothesis (r, p_c) with $r \in \langle E_1, ..., E_m \rangle$ and $|r| < c \leq |r| + m$, then the algorithm outputs a consistent hypothesis. The proof shows that after a

[4] Thus, for this particular algorithm even the worst-case time complexity is polynomial.

particular number of examples (which is polynomial in the parameters n_F and n_G), there almost certainly exists a consistent hypothesis (r, p_c) with $r \in \langle E_1, ..., E_m \rangle$ and $c \in \{|r| + 1, ..., |r| + m\}$, and that any such consistent hypothesis almost certainly correctly predicts the label of the next example. More specifically, we now specify an ERROR-BOUND function of $(\frac{1}{m})^{\frac{1}{2}} = \frac{1}{\sqrt{m}}$ and a DELAY-BOUND function of $m \geq q + (3n_F)^{\frac{4}{3}}$, where q is a constant based on the polynomial p' used in defining the distribution family D_P. (It is worth noting that in this case the DELAY-BOUND does not depend on n_G but on only n_F.) The proof first shows that for $m \geq q + (3n_F)^{\frac{4}{3}}$, the probability (over all possible m-samples drawn randomly, independently according to D_G and labeled by a target drawn randomly according to D_F) that no hypothesis (r, p_c) is consistent, for $r \in \{E_1, E_2, ..., E_m\}$ and $|r| < c < |r| + m$, is at most $\frac{1}{2}(\frac{1}{m})^{\frac{1}{2}} = \frac{1}{2\sqrt{m}}$. The proof then shows that the probability (again relative to D_F and D_G) that a consistent hypothesis (r, p_c) incorrectly labels an example drawn randomly according to D_G is at most $\frac{1}{2}(\frac{1}{m})^{\frac{1}{2}} = \frac{1}{2\sqrt{m}}$. Hence for every $m \geq q + (3n_F)^{\frac{4}{3}}$, the probability (over all targets drawn randomly according to D_F and all $(m+1)$-samples drawn randomly, independently according to D_G) that the hypothesis H_m incorrectly classifies the example x_{m+1} is at most $\frac{1}{2\sqrt{m}} + \frac{1}{2\sqrt{m}} = \frac{1}{\sqrt{m}}$.

Chebyshev's Inequality states that

$$Pr(|Y - \mu| \geq k\sigma) \leq \frac{1}{k^2}$$

where Y is a random variable whose probability distribution has mean μ and standard deviation σ, and $k > 0$. Within Chebyshev's Inequality substitute Y by f and k by $2m^{\frac{1}{4}}$. It follows that for any distribution D_F (with parameter n_F) in D_P, the sum of the probabilities of all possible targets (r, p) such that r does not appear in $\langle E_1, ..., E_m \rangle$ is less than $\frac{1}{4\sqrt{m}}$ when $m \geq \mu + 2\sigma m^{\frac{1}{4}}$. Certainly $m \geq \mu + 2\sigma m^{\frac{1}{4}}$ if $m \geq n_F + 2n_F m^{\frac{1}{4}}$, since $n_F \geq max(\mu, \sigma)$. And certainly $m \geq n_F + 2n_F m^{\frac{1}{4}}$ when $m \geq n_F m^{\frac{1}{4}} + 2n_F m^{\frac{1}{4}} = 3n_F m^{\frac{1}{4}}$, since m is at least 1. Thus for $m \geq (3n_F)^{\frac{4}{3}}$, the sum of the probabilities of all targets (r, p) such that r does not appear in $\langle E_1, ..., E_m \rangle$ is less than $\frac{1}{4\sqrt{m}}$.

Given that the target (r, p) is such that $r \in \langle E_1, ..., E_m \rangle$, what is the probability that $p = p_c$ for $c > |r| + m$? Using the reasoning of the previous paragraph, Chebyshev's Inequality again shows that for $m \geq (3n_F)^{\frac{4}{3}}$, this probability is less than $\frac{1}{4\sqrt{m}}$. Thus the probability (relative to D_F) that the hypothesis space searched by the algorithm does not contain the target is less than $\frac{1}{4\sqrt{m}} + \frac{1}{4\sqrt{m}} = \frac{1}{2\sqrt{m}}$.

The proof is completed by showing that if the hypothesis space searched by the algorithm contains the target, then the probability (over all possible $(m+1)$-samples drawn randomly, independently according to D_G) that the hypothesis H_m incorrectly labels the example x_{m+1} is less than $\frac{1}{2\sqrt{m}}$. It follows from the Blumer Bound that for any ϵ and δ such that

$$\frac{\ln(p'(m)m^2) + \ln\frac{1}{\delta}}{\epsilon} \leq m$$

the hypothesis H_m has the property that with probability at least $1 - \delta$ H_m is $(1 - \epsilon)$-accurate. To ensure that the probability that H_m incorrectly classifies x_{m+1} is less than $\frac{1}{2\sqrt{m}}$, we need simply to choose values for ϵ and δ that sum to $\frac{1}{2\sqrt{m}}$. Choose $\frac{1}{4\sqrt{m}}$ for both ϵ and δ.

Substituting these values into the Blumer bound, we need m such that

$$(\ln(p'(m)m^2) + \ln 4\sqrt{m})4\sqrt{m} \leq m$$

which can be rewritten as

$$\ln(p'(m)m^2) \leq \frac{\sqrt{m}}{4} - \ln 4\sqrt{m}$$

For any particular $p'(m)$ used in defining the distribution family D_P, there exists a least positive integer q such that for $m = q$ this inequality holds. Thus, letting $\text{DELAY-BOUND}(n_F, n_G) = q + (3n_F)^{\frac{4}{3}}$ guarantees that for all $m \geq \text{DELAY-BOUND}(n_F, n_G)$ the probability that H_m incorrectly classifies x_{m+1} is at most $\frac{1}{2\sqrt{m}}$. This completes the proof.

□

Example 9 Time-bounded logic programs are U-learnable under D_P. *Let R be the set of all logic programs that can be built from a given alphabet of predicate symbols $\mathcal{P}$, function symbols $\mathcal{F}$, and variables $\mathcal{V}$. Let k be a positive integer and P^k be the set of all time-bounds $p(x)$ of the form $p(x) = c_i x^k$, where each c_i is a positive integer and x is the size of (number of terms in) the goal. Let the domain X of examples be the ground atoms constructed from $\mathcal{P}$ and $\mathcal{F}$. Let the symbol $\vdash_n$ denote SLDNF derivation time-bounded by n. A concept (r, p) classifies as positive just the atoms $a \in X$ such that $r \vdash_{p(|a|)} a$, where $|a|$ is the number of terms in a. Let the family of distributions D_P be built using some particular enumeration of polynomially-growing subsets of R and some choice of k, and let G be the family of all distributions over examples. From Theorem 8, it follows that (D_P, G) is U-learnable.*

Definition 10 The distribution family D_E. *Let k be a positive integer, and let P^k be the set of univariate polynomial functions of the form $p_{c_j}(x) = c_j x^k$, where c_j is any positive integer. Let T be a finite set of constants, and let D'' be any probability distribution over T. For any $r \in R$, let P_r be the subset of functions $p_c \in P^k$ such that $c = t|r|$ for some $t \in T$. For each exponential size-based distribution D' over R, and each distribution D'' over T, we define a distribution D over all $(r_i, p_{c_j}) \in R \times P^k$ with the following p.d.f.*

$$\begin{aligned} Pr_D(r_i, p_{c_j}) &= 0 \text{ if } c_j \notin P_{r_i} \\ &= [Pr_{D'}(r_i)][Pr_{D''}(t)] \text{ where } c_j = t|r_i| \text{ for } t \in T, \text{ otherwise.} \end{aligned}$$

Let the parameter n of D be the parameter of D'. The distribution family D_E consists of exactly the distributions D defined in this way, for all choices of D' and D''.

Theorem 11 U-learnability under D_E. *Let the distribution family D_E be defined by particular choices of the constants c, k, T, and the enumeration of subsets of R. Let G be any family of distributions over X. The pair of distributions (D_E, G) is U-learnable.*

Proof: Again the proof is constructive. The learning algorithm (Figure 3) is similar to that for D_P, the primary difference being that after any number m of examples, this algorithm considers hypotheses in the sets $E_1, ..., E_{\lfloor\sqrt{m}\rfloor}$ rather than $E_1, ..., E_m$. The proof has much the same form as that for Theorem 8, except that the use of *average-case* time complexity rather than *worst-case* time complexity is crucial. Time complexity is addressed first, followed by correctness.

The U-learning algorithm for D_E
Input: Example sequence $\langle (x_1, l_1), (x_2, l_2), ... \rangle$.
Output: Hypothesis sequence $\langle (r_1, p_1), (r_2, p_2), ... \rangle$.

1. Let $m = 1$.
2. Obtain the next labeled example (x_m, l_m) from the input sequence.
3. Let $i = 1$.
4. If for some $r \in E_i$ and $p_c \in P_r$, the hypothesis (r, p_c) is consistent with the m examples seen thus far, then output (r, p_c) and go to 7.
5. If $i = \lfloor \sqrt{m} \rfloor$ then output any hypothesis and go to 7.
6. Increment i and go to 4.
7. Increment m and go to 2.

Figure 3: The U-learning algorithm for D_E

Time complexity. We must show that there exists a constant b such that for all integers $m \geq 1$ and for all distributions D_G over X, the following sum converges[5] (where L is the learning algorithm for (D_E, G)).

$$\sum_{\text{all } (r, p_d, \vec{X}_m)} [Pr_{D_E, D_G}(r, p_d, \vec{X}_m)] \frac{(\text{TIME}_L(r, p_d, \vec{X}_m))^{\frac{1}{b}}}{M_{(r, p_d, \vec{X}_m)}}$$

We choose $b = 2$. The preceding sum can be rewritten as the following sum.

$$\sum_{k=1}^{\infty} \text{Sum}_k$$

where Sum_k is:

$$\sum_{\{(r, p_d, \vec{X}_m) : r \in E_k, p_d \in P_r\}} [Pr_{D_E, D_G}(r, p_d, \vec{X}_m)] \frac{(\text{TIME}_L(r, p_d, \vec{X}_m))^{\frac{1}{2}}}{M_{(r, p_d, \vec{X}_m)}}$$

The value of Sum_k is bounded by the value of the following sum, where a and a' are appropriately chosen constants, and t' is the largest time-bound coefficient in T.

$$\sum_{\{(r, p_d, \vec{X}_m) : r \in E_k, p_d \in P_r\}} [Pr_{D_E, D_G}(r, p_d, \vec{X}_m)] \frac{[a \sum_{i=1}^{\min(k, \lfloor \sqrt{m} \rfloor)} (c^i a' \sum_{j=1}^{m} p_{t'|r|}(|x_j|))]^{\frac{1}{2}}}{\sum_{j=1}^{m} p_d(|x_j|)}$$

To see that the value of this sum is greater than the value of Sum_k, note that $M_{(r, p, \vec{X}_m)}$ has simply been expressed explicitly as

$$\sum_{j=1}^{m} p_d(|x_j|)$$

[5] Each term in the sum is obviously positive, so to say the sum converges is to say that it is less than infinity.

and $\mathrm{TIME}_L(r,p,\vec{X}_m)$ has been replaced by the upper bound

$$a \sum_{i=1}^{\min(k,\lfloor\sqrt{m}\rfloor)} (c^i a' \sum_{j=1}^{m} p_{t'|r|}(|x_j|))$$

To verify the correctness of this upper bound, notice that L searches through the possible choices of r in $E_1, ..., E_{\lfloor\sqrt{m}\rfloor}$ and all possible choices of time-bound p for a hypothesis (r,p) that is consistent with the examples $\vec{X}_m$. L outputs the *first* hypothesis it finds that is consistent with the examples; thus, if the target is (r,p_d), where $r \in E_k$ and $p_d \in P_r$, and if $k < \lfloor\sqrt{m}\rfloor$, then L will search through at most $E_1, ..., E_k$. As a result, $\mathrm{TIME}_L(r,p,\vec{X}_m)$ is bounded by the sum, for i from 1 to $\min(k,\lfloor\sqrt{m}\rfloor)$, of a constant (a) times the product of:

1. the time to determine whether a given hypothesis in E_i is consistent with $x_1, ..., x_m$, and
2. the number of hypotheses in E_i

Item (1) is bounded by a constant times $\sum_{j=1}^{m} p_{t'|r|}(|x_j|)$. Item (2) is bounded by $|T|c^i$. (We may include both the constant of item (1) and the constant $|T|$ in item (2) in the constant a'.) It follows that $\mathrm{TIME}_L(r,p,\vec{X}_m)$ is bounded by

$$a \sum_{i=1}^{\min(k,\lfloor\sqrt{m}\rfloor)} (c^i a' \sum_{j=1}^{m} p_{t'|r|}(|x_j|))$$

The preceding bound on Sum_k can be rewritten, trivially, as

$$\sum_{\{(r,p):r\in E_k, p\in P_r\}} \mathrm{Sum}_{(r,p)}$$

where $\mathrm{Sum}_{(r,p)}$ is

$$\sum_{\text{all } \vec{X}_m} [Pr_{D_E,D_G}(r,p,\vec{X}_m)] \frac{[a \sum_{i=1}^{\min(k,\lfloor\sqrt{m}\rfloor)} (c^i a' \sum_{j=1}^{m} p_{t'|r|}(|x_j|))]^{\frac{1}{2}}}{\sum_{j=1}^{m} p_d(|x_j|)}$$

Once again using the notation $M_{(r,p,\vec{X}_m)}$, and recognising that $p_{t'|r|}(x)$ is only a constant factor a'' greater than $p_d(x)$, for all x (since d must also be in P_r and therefore is also a constant multiple of $|r|$), it is straightforward to verify that $\mathrm{Sum}_{(r,p)}$ is bounded by the value of the following sum.

$$\sum_{\text{all } \vec{X}_m} [Pr_{D_E,D_G}(r,p,\vec{X}_m)] \frac{[a'' M_{(r,p,\vec{X}_m)} a \lfloor\sqrt{m}\rfloor c^k]^{\frac{1}{2}}}{M_{(r,p,\vec{X}_m)}}$$

Recognising that $M_{(r,p,\vec{X}_m)} \geq \sqrt{m}$ reveals that the value of this sum is in turn bounded by the value of

$$\sum_{\text{all } \vec{X}_m} [Pr_{D_E,D_G}(r,p,\vec{X}_m)] a'' a c^{\frac{k}{2}}$$

which evaluates to $a'' a c^{\frac{k}{2}} Pr_{D_E}(r,p)$. Substituting this value back for $\mathrm{Sum}_{(r,p)}$ in the upper bound on Sum_k, we can rewrite this upper bound to

$$\sum_{\{(r,p):r\in E_k, p\in P_r)=k\}} (a'' a c^{\frac{k}{2}} Pr_{D_E}(r,p))$$

which can be rewritten as

$$a''ac^{\frac{k}{2}} \sum_{\{(r,p):r\in E_k, p\in P_r)=k\}} Pr_{D_E}(r,p))$$

Now the sum of $Pr_{D,E}(r,p)$ over all (r,p) such that $r \in E_k, p \in P_r$ is at most $(c-1)c^{1-k}$, which yields an upper bound of $a''ac^{\frac{k}{2}}(c-1)c^{1-k} = a''a(c-1)c^{1-\frac{k}{2}}$ on Sum_k. Substituting this upper bound into the original sum (in its rewritten form) yields the following upper bound on the original sum.

$$\sum_{k=1}^{\infty} a''a(c-1)c^{1-\frac{k}{2}}$$

This sum can be rewritten as

$$a''a(c-1)\sum_{k=1}^{\infty} c^{1-\frac{k}{2}}$$

which converges just if

$$\sum_{k=1}^{\infty} c^{1-\frac{k}{2}}$$

converges. The latter sum converges just if

$$\lim_{k\to\infty}((c^{1-\frac{k}{2}})^{\frac{1}{k}}) = \lim_{k\to\infty}(c^{\frac{1}{k}})(c^{-\frac{1}{2}}) = c^{-\frac{1}{2}} < 1$$

which is obviously the case since $c > 1$.

Correctness. If $E_1, ..., E_{\lfloor\sqrt{m}\rfloor}$ contains a consistent hypothesis then the algorithm outputs a consistent hypothesis H_m. More specifically, it outputs a consistent hypothesis with the *earliest possible* r. In other words, the algorithm outputs a consistent hypothesis (r_i, p_i) with $r_i \in H_i$ such that: for all $j < i$, any (r_j, p_j) with $r \in H_j$ and $p_j \in P_{r_j}$ is inconsistent. The proof shows that after a particular number of examples (which is polynomial in the parameters n_F and n_G), there almost certainly exists a consistent hypothesis (r,p) with $r \in E_1, ..., E_m$ and $p \in P$, and that any such consistent hypothesis with earliest possible r almost certainly correctly predicts the label of the next example. More specifically, we now specify an ERROR-BOUND function of $(\frac{1}{m})^{\frac{1}{2}} = \frac{1}{\sqrt{m}}$ and a DELAY-BOUND function of $m \geq q + (3n_F)^4$, where q is a constant based on the constant c used in defining the distribution family D_E. The proof first shows that for $m \geq q + (3n_F)^4$, the probability (over all possible m-samples drawn randomly, independently according to D_G and labeled by a target drawn randomly according to D_F) that no hypothesis (r,p) is consistent, for $r \in \langle E_1, E_2, ..., E_{\lfloor\sqrt{m}\rfloor}\rangle$ and $p \in P_r$, is at most $\frac{1}{2}(\frac{1}{m})^{\frac{1}{2}} = \frac{1}{2\sqrt{m}}$. The proof then shows that the probability (again relative to D_F and D_G) that a consistent hypothesis (r,p) with earliest possible r incorrectly labels an example drawn randomly according to D_G is at most $\frac{1}{2}(\frac{1}{m})^{\frac{1}{2}} = \frac{1}{2\sqrt{m}}$. Hence for every $m \geq q + (3n_F)^4$, the probability (over all targets drawn randomly according to D_F and all $(m+1)$-samples drawn randomly, independently according to D_G) that the hypothesis H_m incorrectly classifies the example x_{m+1} is at most $(\frac{1}{m})^{\frac{1}{2}} = \frac{1}{\sqrt{m}}$.

Simply using Chebyshev's Inequality in exactly the way we used it in the proof of Theorem 8, we find that for $\sqrt{m} \geq 3n_F m^{\frac{1}{4}}$, the sum of the probabilities of all targets (r,p) such that r does not appear in $\langle E_1, ..., E_{\lfloor\sqrt{m}\rfloor}\rangle$ is less than $\frac{1}{4\sqrt{m}}$. Thus for $m \geq (3n_F)^4$, the sum of the

probabilities of all targets (r, p) such that r does not appear in $E_1, ..., E_{\lfloor\sqrt{m}\rfloor}$ is less than $\frac{1}{4\sqrt{m}}$, which is less than $\frac{1}{2\sqrt{m}}$, as desired.

The proof is completed by showing that if the hypothesis space searched by the algorithm contains the target, then the probability (over all possible $(m+1)$-samples drawn randomly, independently according to D_G) that the hypothesis H_m incorrectly labels the example x_{m+1} is less than $\frac{1}{2\sqrt{m}}$. Let ES denote

$$\sum_{i=1}^{\lfloor\sqrt{m}\rfloor} Pr_{D_E}(E_i)[|E_1| + ... + |E_i|]$$

It follows from the Average-case Blumer Bound that for any ϵ and δ such that

$$\frac{\ln ES + \ln \frac{1}{\delta}}{\epsilon} \leq m$$

the hypothesis H_m has the property that with probability at least $1 - \delta$ H_m is $(1 - \epsilon)$-accurate. To ensure that the probability that H_m incorrectly classifies x_{m+1} is less than $\frac{1}{2\sqrt{m}}$, we need simply to choose values for ϵ and δ that sum to $\frac{1}{2\sqrt{m}}$. Choose $\frac{1}{4\sqrt{m}}$ for both ϵ and δ. Substituting these values into the Average-case Blumer bound, we need m such that

$$(\ln ES + \ln 4\sqrt{m})4\sqrt{m} \leq m$$

which can be rewritten as

$$\ln ES \leq \frac{\sqrt{m}}{4} - \ln 4\sqrt{m}$$

It can be verified that for any particular choice of c and c' used in defining the distribution family D_P, there exists a least positive integer q such that for $m = q$ this inequality holds.[6] Thus, letting DELAY-BOUND$(n_F, n_G) = q + (3n_F)^4$ guarantees that for all $m \geq$ DELAY-BOUND(n_F, n_G) the probability that H_m incorrectly classifies x_{m+1} is at most $\frac{1}{2\sqrt{m}}$. This completes the proof. □

Example 12 Time-bounded logic programs are U-learnable under D_E. *Let R be the set of all logic programs that can be built from a given finite alphabet of predicate symbols $\mathcal{P}$, function symbols $\mathcal{F}$, and variables $\mathcal{V}$. Let the domain X of examples be the ground atoms built from $\mathcal{P}$ and $\mathcal{F}$. Let the distribution family D_E be built from a choice of $T = \{1, 10, 100, 1000\}$ (to define the allowed time-bounds) and from the enumeration $\langle E_1, E_2, ...\rangle$ of subsets of R such that for all $i \geq 1$ E_i contains all logic programs of length i. Let G be the family of all distributions over examples. From Theorem 11, it follows that (D_E, G) is U-learnable.*

4 Directions for Further Research into Simple U-Learnability

In some cases it may be inappropriate to assume that the probabilities of target representations vanish as quickly as they do in the distributions considered in the preceding theorems. In particular, a more appropriate distribution on target representations in some settings may be one defined

[6] In fact, the same argument can be made using the Blumer Bound rather than the Average-case Blumer Bound, but for distributions such as D_E the Average-case Blumer Bound provides much smaller values for q.

by combining an exponential size-based enumeration as used in Definition 10 (D_E), with some distribution in which the probabilities do not decrease exponentially, as in Definition 7 (some distributions in D_P). U-learnability should be considered with such distributions, perhaps combined with restricted families of distributions on examples and/or restricted representation languages. We expect that obtaining positive results for "natural" distributions of this kind will be theoretically challenging, and that such results will have a major impact on practical applications of machine learning. We also expect that negative results for some such distributions will be challenging and useful. Whereas negative results for PAC-learnability can be obtained by showing that the *consistency* problem for a given concept representation is NP-hard (assuming $RP \neq NP$) [26], negative results for U-learnability in some cases can be obtained by showing that the consistency problem is *NP-hard-on-average* (or *DistNP-hard*) [4, 17] relative to particular distributions on hypotheses and examples. In addition, just as negative results for PAC-predictability can be obtained (based on certain assumptions) using hard problems from cryptography [18], negative results for U-predictability possibly might be obtained in this same way, but would require additional effort.[7] Specifically, obtaining negative results for U-predictability will require specifying a particular distribution (or family of distributions) to be used with a hard problem from cryptography, such as inverting RSA, and it must be verified that the problem does not become substantially easier under this distribution (or every distribution in this family). For example, inverting RSA is assumed hard if encryption/decryption is based on a choice of two large random prime numbers according to a *uniform* distribution over primes of some large size, but it becomes easy if the distribution over primes of a given size assigns probability 1 to one particular prime number, or probability .5 to each of two prime numbers; what about for distributions "in between", where some primes of a given size are more likely than others? The distributions used in U-predictability would correspond to such distributions for RSA.

Another important area in which to search for interesting U-learnability results is in the analysis of existing machine learning algorithms. We conjecture that for every successful inductive learning algorithm there exist interesting, "natural" families of distributions F and G for which that algorithm is a "U-learner". (In other words, if a learning algorithm performs well in practice, this is because it usually receives hypotheses and examples drawn according to distributions for which it is well-suited.) Identifying these distribution families should provide a deeper understanding of these algorithms, their abilities, and their limitations.

5 Extensions to U-learnability

Background knowledge is an essential aspect of many learning problems. The definition of U-learnability does not allow the use of background knowledge. There are at least two natural ways to extend the definition of U-learnability to include the use of background knowledge; this section presents both extensions. The first extension is suited to any representation language, while the second is applicable only to first-order logic or subsets of it. A surprising aspect of the second definition is that it not only captures the notion of inductive learning relative to background knowledge, as desired, but it also provides an alternative to Natarajan's PAC-style formalisation of speed-up learning [25]. Although Natarajan's formalisation is quite appealing, few positive results have been proven within it because it is also very demanding.

The first extension (Section 4.1) is the traditional setting used in Inductive Logic Programming, and its roots go back at least to Plotkin's study of learning logic programs in the limit [28]. It

[7] We thank Micheal Kearns of AT&T Bell Labs for a very helpful general discussion, with one of the authors, related to this point.

assumes that the background theory that is provided is, effectively, a *syntactic part* of the target concept. When applied to logic programs, one implication of this is that the background theory entails, or implies, a subset of the ground atoms that the target concept entails. But there are other implications of this as well. Most importantly, the learner never needs to *modify* the given background theory, but only to add to it. The second extension (Section 4.2) assumes only that the background theory implies a subset of the expressions in the domain X (for example, ground atoms) that the target implies. Notably, this extension does not assume any syntactic relationship between the background theory and the target. Thus, it is likely that the learning algorithm will need to modify the background theory rather than merely add to it. Therefore, this extension is perhaps more realistic, and it is certainly more appropriate to speed-up learning and knowledge-base refinement.

5.1 U-learnability with background knowledge (standard setting)

5.1.1 Preliminaries

Let $(\Sigma_E, X, \Sigma_C, R, \Sigma_B, B, c)$ be the representation of a learning problem with background knowledge, where X is the domain of examples (finite strings over the alphabet Σ_E), R is the class of concept representations (finite strings over the alphabet Σ_C), B is the class of background theories (finite strings over the alphabet Σ_B), and $c : R \times B \rightarrow 2^X$ maps any concept representation together with any background theory to a concept (subset of X).

Let P, a subset of the polynomial functions in one variable, be a set of time-bounds. A concept representation is a triple (r, b, p), for $r \in R$, $b \in B$, and $p \in P$. Let $A(r, b, p, x)$ be an evaluation algorithm that maps any tuple (r, b, p, x), for $r \in R$, $b \in B$, $p \in P$, and $x \in X$, to 0 or 1. Let A run in time bounded by $p(|x|)$. Furthermore, let A have the following properties. First, for all $r \in R$, $b \in B$, and $x \in X$, if $x \notin c(r, b)$ then for every $p \in P$: $A(r, b, p, x) = 0$. Second, for all $r \in R$, $b \in B$, and $x \in X$, if $x \in c(r, b)$ then there exists $p \in P$ such that for every $p' \in P$ with $p'(|x|) \geq p(|x|)$: $A(r, b, p', x) = 1$.

Let F be any family of probability distributions over $R \times B \times P$, where the distributions in F each have an associated parameter $n_F \geq 0$ (n_F might be the greater of the mean and standard deviation of each distribution). Let G be any family of probability distributions over X, where the distributions in G each have an associated parameter $n_G \geq 0$.

5.1.2 Protocol

In U-learning with background knowledge, a Teacher randomly chooses a target (r, b, p), for $r \in R$, $b \in B$, and $p \in P$, according to some distribution D_F in F. The Teacher provides the learning algorithm L with b. The Teacher then presents to L an infinite stream of labeled examples $\langle (x_1, l_1), (x_2, l_2), ... \rangle$ where: each example x_i is drawn randomly, independently of the preceding examples, from X according to some fixed, unknown distribution D_G in G and labeled by $l_i = A(r, b, p, x_i)$. After each example x_i in the stream of examples, L outputs a hypothesis $H_i = (r_i, p_i)$, where $r_i \in R$ and $p_i \in P$.

5.1.3 Definition of U-learnability with background knowledge (standard setting)

Definition 13 U-learnability with background knowledge (standard setting). *Let F and G be families of distributions over $R \times B \times P$ and X respectively. The pair (F,G) is* U-learnable *just if there exist a learning algorithm L and three polynomial functions,* LEARNTIME-BOUND(x),

The U-learning algorithm for D_E with background knowledge
Input: Background Theory b and Example sequence $\langle (x_1, l_1), (x_2, l_2), \ldots \rangle$.
Output: Hypothesis sequence $\langle (r_1, p_1), (r_2, p_2), \ldots \rangle$.

1. Let $m = 1$.
2. Obtain the next labeled example (x_m, l_m) from the input sequence.
3. Let $i = 1$.
4. If for some $r \in E_i$ and $p_c \in P_{r,b}$ (where b is the same as the background theory provided as input), the hypothesis (r, b, p_c) is consistent with the m examples seen thus far, then output (r, p_c) and go to 7.
5. If $i = \lfloor \sqrt{m} \rfloor$ then output any hypothesis and go to 7.
6. Increment i and go to 4.
7. Increment m and go to 2.

Figure 4: The U-learning algorithm for D_E with background knowledge

DELAY-BOUND(x, y), *and* ERROR-BOUND(x), *such that for every distribution D_F (with parameter n_F) in F, and every distribution D_G (with parameter n_G) in G, the following hold.*

Time Complexity. *The average-case time complexity of L at any point in a run is bounded by* LEARNTIME-BOUND(M), *where M is the sum of $p(|x_i|)$ over the examples x_i seen to that point.*

Correctness. *For all $m \geq$* DELAY-BOUND(n_F, n_G),

$$\sum_{all\ (r,b,p,\vec{X}_{m+1})} [Pr_{D_F,D_G}(r, b, p, \vec{X}_{m+1})][E_L(r, b, p, \vec{X}_{m+1})] < \text{ERROR-BOUND}(\frac{1}{m})$$

where $E_L(r, b, p, \vec{X}_{m+1}) = 0$ if L correctly classifies example x_{m+1} given examples $\langle x_1, ..., x_m \rangle$ labeled according to (r, b, p), and $E_L(r, b, p, \vec{X}_{m+1}) = 1$ otherwise.[8]

It is straightforward to verify that each of Theorem 8 and Theorem 11 holds when background theories are incorporated in the obvious way. The learning algorithm for D_E with background information is presented in Figure 4.

Example 14 Time-bounded logic programs with background knowledge are learnable under D_E. *Let R and B be the set of all logic programs that can be built from a given finite alphabet of predicate symbols $\mathcal{P}$, function symbols $\mathcal{F}$, and variables $\mathcal{V}$. And again let the domain X of examples be the ground atoms built from $\mathcal{P}$ and $\mathcal{F}$. Let the distribution family D_E be built from the enumeration $\langle E_1, E_2, ... \rangle$ of subsets of $R \times B$ such that for all $i \geq 1$ E_i contains all pairs (r, b) such that $|r| + |b| = i$, as well as from a particular choice of a set of constants T used in specifying the time-bounds $P_{r,b}$ for each value of $|r| + |b|$. Let G be the family of all distributions over examples. The algorithm in Figure 4 U-learns (D_E, G) with background knowledge. In fact, in*

[8] We take $Pr_{D_F,D_G}(r, b, p, \vec{X}_{m+1})$ to be the product of (1) the probability of (r, b, p) according to D_F and (2) the probability of drawing $\vec{X}_{m+1}$ when drawing randomly and independently an $(m+1)$-sample according to D_G.

the case where r is a single definite clause, this learning algorithm can be viewed as an idealisation of the program Progol *that already has been successfully applied to two significant scientific domains as well as a number of standard benchmark problems for machine learning [34].*

5.2 U-learnability with background knowledge (with re-expression of background knowledge)

5.2.1 Preliminaries

Let $(\Sigma_E, X, \Sigma_C, R, B, c)$ be the representation of a learning problem with background knowledge, where X is the domain of examples (first-order expressions over the alphabet Σ_E), R is the class of concept representations (first-order expressions over the alphabet Σ_C), $B \subseteq R$ is the class of background theories, and $c : R \rightarrow 2^X$ maps any concept representation r to a concept (subset of X) consisting of exactly the expressions in X that are entailed, or logically implied, by r. We say that $b \in B$ is consistent with $r \in R$ just if r entails b. We assume that both R and B contain the empty theory.

Let P, a subset of the polynomial functions in one variable, be a set of time-bounds. A concept representation is a pair (r, p), for $r \in R$ and $p \in P$. Let $A(r, p, x)$ be an evaluation algorithm as given in the original definition of U-learnability (without background knowledge).

Let F be any family of probability distributions over $R \times P$, where the distributions in F each have an associated parameter $n_F \geq 0$. Let G be any family of probability distributions over X, where the distributions in G each have an associated parameter $n_G \geq 0$.

5.2.2 Protocol

A Teacher randomly chooses a target (r, p), for $r \in R$ and $p \in P$, according to some distribution D_F in F. The Teacher provides the learning algorithm L with some background theory $b \in B$ that is consistent with r. The Teacher then presents to L an infinite stream of labeled examples $\langle (x_1, l_1), (x_2, l_2), ... \rangle$ where: each example x_i is drawn randomly, independently of the preceding examples, from X according to some fixed, unknown distribution D_G in G and labeled by $l_i = A(r, p, x_i)$. After each example x_i in the stream of examples, L outputs a hypothesis $H_i = (r_i, p_i)$, where $r_i \in R$ and $p_i \in P$.

5.2.3 Definition of U-learnability with background knowledge (with re-expression of background knowledge)

The definition of U-learnability remains the same as in the standard setting with background knowledge. The difference is simply in the protocol, that the background theory b may need to be re-expressed in order to be used efficiently (i.e., with a time-bound not too different from that used by the target).

Note that in the special case where b is logically equivalent to r we have the traditional setting for speed-up learning. The learner already has the *logical* definition of the target concept, but lacks the appropriate time bound and *efficient* representation of the target concept that is being used by the teacher. We suggest that this definition of U-learnability as a formalisation of speed-up learning will yield more positive results than has the use of Natarajan's formalisation, and that these results will be useful in practice.

6 Conclusion

Powerful representation languages, such as Horn clause logic and decision trees, often appear to be tractable in real-world domains. This seems to fly in the face of negative or missing PAC-learnability results for these representations. The apparent contradiction may be explained by the fact that very different distributional assumptions are made by various learning algorithms or for various applications, and the ability to make and to change these assumptions allows an apparently intractable representation to be used efficiently.

The choice of representation is at the centre of focus in PAC-learning. By contrast, the choice of distribution is the central theme in U-learnability. The reason for the refocusing of attention is that once one is committed to a Universal representation, for example as in the case of *inductive logic programming*, differences in distributional assumptions become paramount.

Some general initial results of U-learnability are stated in this paper. Much more effort is required to clearly define the boundaries of what is and is not U-learnable. We expect that this direction of research will yield results with both theoretical and practical significance.

Acknowledgements

The authors would like to thank Tony Hoare, Donald Michie, and Ashwin Srinivasan of Oxford University Computing Laboratory for discussions and input to this paper. This work was supported partly by the Esprit Basic Research Action ILP (project 6020), the SERC project GR/J46623 and an SERC Advanced Research Fellowship held by the first author. The first author is also supported by a Research Fellowship at Wolfson College, Oxford.

References

[1] D. Angluin and M. Kharitonov. When won't membership queries help? In *Proceedings of the 23rd ACM Symposium on Theory of Computing*, pages 444–454. ACM, 1991.

[2] Martin Anthony, Norman Biggs, and John Shawe-Taylor. The learnability of formal concepts. In *Proceedings of the 1990 Workshop on Computational Learning Theory*, pages 246–257, Rochester, NY, August 1990. Morgan Kaufmann.

[3] B. Arbab and D. Michie. Generating rules from examples. In *IJCAI-85*, pages 631–633, Los Altos, CA, 1985. Kaufmann.

[4] S. Ben-David, B. Chor, O. Goldreich, and M. Luby. On the theory of average case complexity. *Journal of Information and System Sciences*, 44:193–219, 1992.

[5] A.W. Biermann. The inference of regular LISP programs from examples. *IEEE Transactions on Systems, Man and Cybernetics*, 8(8):585–600, 1978.

[6] A.W. Biermann and R. Krishnaswamy. Constructing programs from example computations. *IEEE Transactions on Software Engineering*, 2(3), 1976.

[7] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Occam's razor. *Information Processing Letters*, 24(6):377–380, 1987.

[8] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, 36(4):929–965, 1989.

[9] I. Bratko. Generating human-understandable decision rules. Working paper, E. Kardelj University Ljubljana, Ljubljana, Yugoslavia, 1983.

[10] W. Buntine. *A Theory of Learning Classification Rules.* PhD thesis, School of Computing Science, University of Technology, Sydney, 1990.

[11] P. Cheeseman, J. Kelly, M. Self, J. Stutz, W. Taylor, and D. Freeman. AutoClass: a bayesian classification system. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 54–64, San Mateo, CA, 1988. Morgan Kaufmann.

[12] P. Cheeseman, M. Self, J. Kelly, W. Taylor, D. Freeman, and J. Stutz. Bayesian classification. In *Proceedings of the National Conference on Artificial Intelligence (AAAI88)*, pages 607–611, San Mateo, CA, 1988. Morgan Kaufmann.

[13] W. W. Cohen. Cryptographic limitations on learning one-clause logic programs. In *AAAI-93*, Menlo Park, CA, 1993. AAAI Press.

[14] J. Fisher. *Statistical Methods for Research Workers.* Oliver and Boyd, London, 1970.

[15] E. M. Gold. Language identification in the limit. *Inf. Control*, (10):447–474, 1967.

[16] D. Haussler, M Kearns, and R. Shapire. Bounds on the sample complexity of bayesian learning using information theory and the vc dimension. In *COLT-91: Proceedings of the 4th Annual Workshop on Computational Learning Theory*, pages 61–74, San Mateo, CA, 1991. Morgan Kauffmann.

[17] D. Johnson. The NP-completeness column—an ongoing guide. *Journal of Algorithms*, 4:284–299, 1984.

[18] M. Kearns and L. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. In *Proceedings of the 21st ACM Symposium on Theory of Computing*, pages 433–444. ACM, 1989.

[19] R. King, S. Muggleton R. Lewis, and M. Sternberg. Drug design by machine learning: The use of inductive logic programming to model the structure-activity relationships of trimethoprim analogues binding to dihydrofolate reductase. *Proceedings of the National Academy of Sciences*, 89(23), 1992.

[20] R. King and M.J.E. Sternberg. A machine learning approach for the prediction of protein secondary structure. *Journal of Molecular Biology*, 216:441–457, 1990.

[21] R. S. Michalski. A theory and methodology of inductive learning. In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 82–132. Morgan Kaufmann Publishers, 1983.

[22] D. Michie, D. Spiegelhalter, and C. Taylor. *Machine Learning, Neural, and Statistical Classification.* Ellis Horwood Limited, New York, 1994.

[23] S. Muggleton, R. King, and M. Sternberg. Protein secondary structure prediction using logic-based machine learning. *Protein Engineering*, 5(7):647–657, 1992.

[24] S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 12, 1994. (to appear).

[25] B. K. Natarajan. Learning from exercises. In *Proceedings of the 1989 Workshop on Computational Learning Theory*, pages 72–86, San Mateo, CA, 1989. Morgan Kaufmann.

[26] L. Pitt and L. G. Valiant. Computational limitations on learning from examples. *Journal of the ACM*, 35(4):965–984, 1988.

[27] L. Pitt and M. Warmuth. Prediction-preserving reducibility. *Journal of Computer and System Sciences*, 41:430–467, 1990.

[28] G.D. Plotkin. *Automatic Methods of Inductive Inference*. PhD thesis, Edinburgh University, August 1971.

[29] J. Rissanen. Modeling by Shortest Data Description. *Automatica*, 14:465–471, 1978.

[30] J. Rissanen. A universal prior for integers and estimation by Minimum Description Length. *Annals of Statistics*, 11:416–431, 1982.

[31] C. Sammut, S. Hurst, D. Kedzier, and D. Michie. Learning to fly. In D. Sleeman and P. Edwards, editors, *Proceedings of the Ninth International Workshop on Machine Learning*, pages 385–393, San Mateo, CA, 1992. Morgan Kaufmann.

[32] E. Y. Shapiro. *Algorithmic Program Debugging*. MIT Press, Cambridge, MA, 1983.

[33] R.J. Solomonoff. A formal theory of inductive inference. *Information and Control*, 7:376–388, 1964.

[34] A. Srinivasan, S. Muggleton, R. King, and M. Sternberg. Mutagenesis: Ilp experiments in a non-determinate biological domain. Submitted to the Fourth Workshop on Inductive Logic Programming, 1994.

[35] M. Sternberg, R. King, R. Lewis, and S. Muggleton. Application of machine learning to structural molecular biology. *Philosophical Transactions of the Royal Society B*, 1994 (to appear).

[36] M. Sternberg, R. Lewis, R. King, and S. Muggleton. Modelling the structure and function of enzymes by machine learning. *Proceedings of the Royal Society of Chemistry: Faraday Discussions*, 93:269–280, 1992.

[37] P.D. Summers. *Program construction from examples*. PhD thesis, Yale University, New Haven, CT, 1975.

[38] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.