Reprinted from INFORMATION AND COMPUTATION
All Rights Reserved by Academic Press, New York and London

Vol. 95, No. 2, December 1991
Printed in Belgium

Equivalence of Models for Polynomial Learnability*

DAVID HAUSSLER

Baskin Center for Computer Engineering and Information Sciences, University of California at Santa Cruz, Santa Cruz, California 95064

MICHAEL KEARNS†

Aiken Computation Laboratory, Harvard University, Cambridge, Massachusetts 02138

AND

NICK LITTLESTONE‡ AND MANFRED K. WARMUTH

Baskin Center for Computer Engineering and Information Sciencies, University of California at Santa Cruz, Santa Cruz, California 95064

In this paper we consider several variants of Valiant's learnability model that have appeared in the literature. We give conditions under which these models are equivalent in terms of the polynomially learnable concept classes they define. These equivalences allow comparisons of most of the existing theorems in Valiant-style learnability and show that several simplifying assumptions on polynomial learning algorithms can be made without loss of generality. We also give a useful reduction of learning problems to the problem of finding consistent hypotheses, and give comparisons and equivalences between Valiant's model and the prediction learning models of Haussler, Littlestone, and Warmuth (*in* "29th Annual IEEE Symposium on Foundations of Computer Science," 1988).

1. INTRODUCTION

The model introduced by Valiant (1984) provides the framework for a growing body of research in machine learning (Blumer *et al.*, 1987, 1989; Kearns *et al.*, 1987; Pitt and Valiant, 1988; Rivest, 1987; Valiant, 1984, 1985). This research focuses on understanding the computational complexity of various learning tasks. A central notion is that of polynomial learnability. Rougly speaking, a concept class is said to be polynomially

* The authors gratefully acknowledge the support of ONR Grant N00014-86-K-0454. M. Kearns was also supported by an AT & T Bell Laboratories Scholarship. This research was done while M. Kearns was visiting the University of California at Santa Cruz.

† Current address: AT & T Bell Laboratories, Murray Hill, New Jersey 07974.

‡ Current address: NEC Research Institute, 4 Independence Way, Princeton, New Jersey 08540.

0890-5401/91 $3.00

learnable if there exists an algorithm that can find a hypothesis approximating any concept in the class,[1] when given a polynomial number of examples of the concept and polynomially bounded computational resources. The polynomial growth is with respect to parameters measuring the complexity of the concept, the size of the input to the algorithm, and the accuracy of the resulting approximation.

The specific assumptions and criteria used to define polynomial learnability have varied among the different researchers in the field. To allow confident and accurate comparisons of the results obtained in the different models, it is important to verify the equivalence of the models, or to discover any differences. We perform that task here. The result is a unification of previous work, a precise definition of polynomial learnability, and an understanding of variations in the model that do not affect what is polynomially learnable. A significant part of this paper consists of formal proofs of "folk theorems" that have been known to many researchers in computational learning theory for some time, but have never been documented systematically.

In the first part of this paper (Sections 2 and 3), we consider a number of existing variations of the learning model and show that they lead to equivalent models of polynomial learnability. Some of the equivalence proofs formalize arguments made informally in private communications to the authors; others are new. We show that if all other parameters of the model are equal, then the model where algorithms have access to a single oracle returning labeled examples is equivalent to the model where there are two oracles returning positive and negative examples, respectively; that a model where a learning algorithm must output a good hypothesis with only a fixed probability is equivalent to one where it must do so with arbitrarily high probability; and that without loss of generality, all learning algorithms can be deterministic.

In the process of formalizing the equivalences, we have uncovered interactions between the initial information given to an algorithm and its ability to halt deterministically in all cases. Our results demonstrate that the models used in Blumer *et al.* (1987, 1989), Kearns *et al.* (1987), Pitt and Valiant (1988), Rivest (1987), and Valiant (1984, 1985) are equivalent if probabilistic halting criteria are substituted for deterministic halting in some of the models. The equivalence if deterministic halting is required remains an open question. The results also show the equivalence of other natural variations of these models in nearly all possible combinations of the various modifications that we consider. In only one equivalence proof (the one that shows that the one- and two-oracle models are equivalent) do we

[1] This model has been termed the *PAC-model* by Angluin (1987) standing for *probably approximately correct.*

actually change the distribution on the examples. Thus most of the proven equivalences hold also for learning with respect to any fixed distribution on the examples.

After demonstrating these equivalences in Section 3, we turn to different types of equivalence in Sections 4 and 5. In Blumer *et al.* (1989), a necessary condition for polynomial learnability is given in terms of a combinatorial parameter of the concept class called the Vapnik–Chervonenkis dimension (Vapnik and Chervonenkis, 1971). This condition does not address issues of computational complexity but does address the minimum amount of information needed to perform inductive inference. Using tools from Pitt and Valient (1988), we show that the problem of polynomially learning C by H is equivalent (modulo polynomial-time transformations) to the problem of polynomially finding (with fixed probability) a hypothesis in H consistent with a given sequence of examples of a target concept in C, provided that the VC dimension of H grows only polynomially with the concept complexity measure used for C. This allows one in many cases to view a learning problem in the more traditional light of complexity-theoretic search problems.

The definitions of learnability discussed here depend fundamentally on the hypothesis space from which the learning algorithm must choose its hypothesis. In many cases it is of interest to study the learnability of a concept class when no restriction is placed on the hypothesis space to be used. For this purpose, a new model of learning was introduced in Haussler *et al.* (1988) that discards the constraints placed on hypotheses of the learning algorithms. In that model the polynomial learning algorithm must predict accurately the label (positive or negative) of an unlabeled random example after receiving polynomially many random examples that are labeled consistently with the target concept. In the main theorem of Section 5 we show that a concept class is polynomially learnable in the prediction model of Haussler *et al.* (1988) iff there exists a polynomially evaluatable (defined in the next section) hypothesis space such that the concept class is polynomially learnable by this hypothesis space.

2. Models of Polynomial Learnability

2.1. *Representing Examples and Concepts*

Definition 2.1. Representation of domains. For each $n \geq 1$, X_n denotes a set called a *learning domain* on n attributes. The X_n for different values of n are assumed to be disjoint. We let $\mathbf{X} = \{X_n\}_{n \geq 1}$. We say a *point* (or *instance*) x is in $\mathbf{X}$ if $x \in \bigcup \{X_n\}_{n \geq 1}$.

For example, X_n might be the set of all points in Euclidean n-dimensional space $\mathbf{R}^n$ or the Boolean domain $\{0, 1\}^n$.[2]

For the purposes of computation, we assume that points are encoded as tuples using any of the standard schemes (see Garey and Johnson, 1979) in such a way that the representation of each point in X_n has length between n and $l = l(n)$, where $l(n)$ is a polynomial. The equivalence results given below will not depend on how real numbers are handled (i.e., whether the uniform or logarithmic cost model is used to define the length of representations and inputs to algorithms (Aho *et al.*, 1974). We assume that from the representation of any tuple of $\mathbf{X}$ one can efficiently determine the unique set X_n to which it belongs.

We think of concepts as subsets of a learning domain X_n. For some purposes, we must also have a language in which concepts are represented as strings, and a notion of the size or complexity of a concept, which is usually related to the length of its shortest representation.

DEFINITION 2.2. Representations of concepts and concept classes. Given a learning domain X, a set $C \subseteq 2^X$ is called a set of *concepts* on X. A *representation* for C consists of a set of strings L and a mapping σ from L onto C that associates each string in L with a concept in C. A *concept complexity measure* for C is a mapping **size** from C to $\{1, 2, ...\}$.

For each $n \geq 1$, let $C_n \subseteq 2^{X_n}$ be a set of concepts, L_n and σ_n be a representation for C_n, and $\mathbf{size}_n$ be a concept complexity measures for C_n. Then $\mathbf{C} = \{(X_n, C_n, L_n, \sigma_n, \mathbf{size}_n)\}_{n \geq 1}$ denotes a *concept class* over $\mathbf{X}$. Normally the representation and concept complexity measure will be understood from the context, in which case we will abbreviate $\mathbf{C}$ as $\{(X_n, C_n)\}_{n \geq 1}$. We say that a concept c is in $\mathbf{C}$ if $c \in \bigcup C_n$. When the concept complexity measure is understood from the context, it is also convenient to let $C_{n,s}$ denote $\{c \in C_n : \mathbf{size}(c) \leq s\}$.

To illustrate these definitions, consider the concept class k-CNF from Valiant's original paper (1984) for some fixed $k \geq 1$. Here $X_n = \{0, 1\}^n$, the representation language L_n consists of all CNF expressions on n variables (say $x_1, ..., x_n$) that have at most k literals per clause, C_n consists of all $c \subseteq \{0, 1\}^n$ such that c is the set of satisfying assignments of one of these expressions, σ_n maps a CNF expression to its set of satisfying assignments, and $\mathbf{size}_n(c)$ is the number of literals in the smallest k-CNF representation of c. The concept classes k-DNF, k-term DNF, and k-clause CNF are defined similarly by restricting to DNF expressions with at most k literals per term, DNF expressions with at most k terms, and CNF expressions with at most k clauses, respectively. The concept classes DNF and CNF

[2] We assume that any learning domain X_n that we consider can be embedded in $\mathbf{R}^n$ by a measurable embedding.

can also be defined in this manner. Here $C_n = 2^{\{0,1\}^n}$, but $C_{n,s}$ includes only those concepts that can be represented by a DNF (resp. CNF) expression with at most s literals.

For real-valued domains, examples of concept classes include the class of closed halfspaces and the class of closed convex polytopes. In the case of halfspaces in n dimensions, one possibility is to assume that concepts are represented by the $n+1$ coefficients of the separating hyperplane, and that $\mathbf{size}_n(c) = n+1$ for all $c \in C_n$ (uniform cost model). In the case of convex polytopes, concepts might be represented either as the intersection of a set of halfspaces, or by specifying the vertices of the convex polytope.

DEFINITION 2.3. Examples and hypotheses. Let **C** be a concept class over **X**. Given a concept c in **C**, we define an *example* of c to be a pair $\langle x, a \rangle$, where x is in **X** and $a \in \{0, 1\}$ such that $a = 1$ iff $x \in c$. If $x \in c$ then $\langle x, 1 \rangle$ is called a *positive* example of c, and if $x \notin c$ then $\langle x, 0 \rangle$ is called a *negative* example of c. A *sample* of c is a sequence of examples of c. As above, for computational purposes we assume samples are represented as sequences of pairs using any of the standard schemes. The *size* of a sample is the number of examples it contains. The *length* of a sample is the number of symbols in its encoding. (In case of the uniform cost model, each real number contributes one to the length.)

Let **C** and **H** be two concept classes. An algorithm for learning **C** by **H** is an algorithm that when given examples of some concept $c \in \mathbf{C}$ will produce as output (a representation of) some concept $h \in \mathbf{H}$ that is an approximation of c in a sense made precise below. The class **C** is called the *target class* and c is called the *target concept*. The class **H** is called the *hypothesis* space used by the algorithm and the hypothesis h (whose representation is output by the algorithm) is called the *hypothesis* of the algorithm.

We say that a hypothesis h in **H** is *consistent* with a sample $\langle x_1, a_1 \rangle, \ldots, \langle x_m, a_m \rangle$ of c if $x_i \in h \Leftrightarrow a_i = 1$ for all $1 \leqslant i \leqslant m$.

2.2. *Models for Polynomial Learnability*

We now define the three most popular variants of Valiant's original model. In the following models, we assume that **X** is a learning domain and that **C** and **H** are concept classes over **X**.

Model 1. The Functional Model. In this model a learning algorithm implements a function that maps from samples to hypotheses. If $c \in C_n$ is a given target concept, $h \in H_n$ is a hypothesis, and D is any fixed probability distribution on X_n, then define the *error* of h (with respect to c and D) to be the probability that h is inconsistent with a random example of c (i.e., an example $\langle x, a \rangle$ of c in which x is drawn randomly

from X_n according to D). We will say that **C** is *polynomially learnable by* **H** (*in the functional model*) if there exists an algorithm A that takes as input a sample of a target concept in **C**, and outputs a representation of a hypothesis in **H** such that the following property holds:

Property 1. (a) There is a function $m(\varepsilon, \delta, n, s)$, polynomial in $1/\varepsilon$, $1/\delta$, n and s, such that for all $0 < \varepsilon, \delta < 1$, and $n, s \geq 1$, and for all target concepts $c \in C_{n,s}$ and all probability distributions D on X_n, if A is given a random sample of c of at least $m(\varepsilon, \delta, n, s)$ examples drawn independently according to D, then A produces a representation in **H** of a hypothesis $h \in H_n$, and with probability at least $1 - \delta$ the hypothesis h has error at most ε.

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

We let $S_A(\varepsilon, \delta, n, s)$ denote the smallest sample size $m(\varepsilon, \delta, n, s)$ such that Property 1(a) holds for algorithm A. $S_A(\varepsilon, \delta, n, s)$ is called the *sample complexity* of A. Note that in the functional model the algorithm A is not given any of the parameters ε, δ, n, and s as input. The only input to A is a batch of examples. Throughout the paper ε will be called the *accuracy parameter* and δ the *confidence parameter*.

Model 2. The One-Oracle Model. Instead of specifying that A be simply a function mapping samples to hypotheses, we can allow A to explicitly use information about the desired accuracy and confidence parameters ε and δ, as well as the complexity parameters n and s. This can be accomplished by giving ε, δ, n, and s as input to A and supplying A with an oracle EX for random examples of the target concept. Each time EX is called, it selects an instance in X_n independently at random according to the distribution D and returns it along with a label indicating whether or not it is in the target concept. Throughout this paper whenever an oracle returns an example to an algorithm, then we charge the algorithm with time equal to the length of the received example. We say that **C** *is polynomially learnable by* **H** (*in the one-oracle model*) if there is an algorithm A taking inputs ε, δ, n, and s and outputting a representation of a hypothesis in **H** such that the following property holds:

Property 2. For all $0 < \varepsilon, \delta < 1$, and $n, s \geq 1$, and for all target concepts $c \in C_{n,s}$ and all probability distributions D on X_n,

(a) A outputs a representation in **H** of a hypothesis $h \in H_n$, and with probability at least $1 - \delta$ the output hypothesis h has error at most ε.

(b) The total running time is bounded by a polynomial in $1/\varepsilon$, $1/\delta$, n, and s.

In this model, the sample complexity $S_A(\varepsilon, \delta, n, s)$ of an algorithm A is taken to be the worst-case number of oracle calls on input ε, δ, n, s, over

all $c \in C_{n,s}$ and all sequences of examples of c. Similarly, the worst-case running time is denoted $T_A(\varepsilon, \delta, n, s)$.

Model 3. The Two-Oracle Model. The one-oracle model can be further modified to allow the algorithm A access to two oracles, one that returns random positive examples of the target concept (which we will call *POS*), and one that returns random negative examples (which we will call *NEG*). In this case there are two distributions D^+ and D^-. D^+ is a distribution on c and D^- is a distribution on $X_n - c$. Calls to *POS* return examples chosen according to D^+ and calls to *NEG* return examples chosen according to D^-.

In this two-oracle model we also define two types of error: $error^+$ (the *positive error*) is the probability that a random example form *POS* is classified as negative by the hypothesis, and $error^-$ (the *negative error*) is the probability that a random example from *NEG* is classified as positive by the hypothesis. The definition of polynomial learnability is now as in the one-oracle model, except that in Property 2 we now require the hypothesis of learning algorithm A to have both positive error at most ε and negative error at most ε, that is:

Property 3. For all $0 < \varepsilon, \delta < 1$ and $n, s, \geqslant 1$, and for all target concepts $c \in C_{n,s}$ and all probability distributions D on X_n,

(a) A outputs a representation in **H** of a hypothesis $h \in H_n$, and with probability at least $1 - \delta$ the output hypothesis h has positive error at most ε and negative error at most ε.

(b) The total running time is bounded by a polynomial in $1/\varepsilon$, $1/\delta$, n and s.

The main contribution of this paper is to prove the equivalence of these models, along with a number of additional variations. The equivalence results depend only on weak assumptions about the target class and the hypothesis space, outlined below. The variations that we consider are:

(1) Randomized vs. deterministic: We consider both randomized and deterministic learning algorithms. Randomized algorithms are allowed to make use of flips of a fair coin. These coin flips are independent of the random examples of the target concept received by the algorithm. A randomized algorithm is charged one unit of time for each coin-flip that it uses. The sample complexity $S_A(\varepsilon, \delta, n, s)$ and the running time $T_A(\varepsilon, \delta, n, s)$ are extended to worst case measures over the coin-flips of the algorithm. We show that without loss of generality, with respect to polynomial learnability all learning algorithms are deterministic (modulo some weak regularity assumptions on the hypothesis space that we discuss below).

(2) Polynomial in $1/\delta$ vs. polynomial in $\log(1/\delta)$ vs. fixed δ: We consider three ways to treat the confidence parameter δ. In one case the sample and time complexities of a polynomial learning algorithm are required to be polynomial in $1/\delta$, as described above. In the second case, the sample and time complexities are required to be polynomial in $\log 1/\delta$. In the final case we fix $\delta = \delta_0$ for any constant $0 < \delta_0 < 1$ (so the learning algorithm no longer requires δ as an input), and we require only that a polynomial learning algorithm achieve this level of confidence, with sample and time complexity polynomial in the remaining variables of the model. We show that in all three cases, the same class of polynomially learnable concept classes is obtained.

For the one-oracle and two-oracle models, we also consider the case in which the value of s is not given as input to the algorithm. We will show that such a model is equivalent to the model where s is given, provided that one also relaxes the halting criterion, changing it to a probabilistic halting criterion. Similar results are obtained in Linial *et al.* (1988) (see also Benedek and Itai (1988b). We thus consider the following two further modifications:

(3) Knowledge of s vs. no knowledge of s.

(4) Deterministic halting vs. probabilistic halting. For the probabilistic halting case, property 2 in the definitions of the one-oracle model is changed to:

Property 2′. There exists some polynomial p such that for all $0 < \varepsilon$, $\delta < 1$, and $n, s \geq 1$, and for all target concepts $c \in C_{n,s}$ and all probability distributions D on X_n, with probability at least $1 - \delta$,

(a) A outputs a representation in **H** of a hypothesis in H_n that has error at most ε.

(b) The total running time is bounded by $p(1/\varepsilon, 1/\delta, n, s)$.

Property 3 in the definition of the two-oracle model is changed analogously. Sample complexity is not defined in the probabilistic halting case. Note that Property 2′ allows for the possibility that the expected running time of the learning algorithm is infinite.

For convenience of notation, we will introduce a parameterized version of the three basic models along with their various modifications. Thus, **functional**(p_1, p_2) will denote those pairs of concept classes (**C**, **H**) such that **C** is polynomially learnable by **H** in the functional model under modifications p_1 and p_2, where p_1 is either $1/\delta$ (when polynomial dependence on $1/\delta$ allowed), $\log(1/\delta)$ (for restriction to polynomial dependence on $\log(1/\delta)$), or *fixed* δ (when we fix δ to $0 < \delta_0 < 1$), and p_2 is either *rand* (for randomized learning algorithms) or *det* (for deterministic algo-

rithms). Similarly, **one-oracle**(p_1, p_2, p_3, p_4) and **two-oracle**(p_1, p_2, p_3, p_4) will denote those pairs of concept classes (**C**, **H**) such that **C** is polynomially learnable by **H** in the one- and two-oracle models respectively, under modifications p_1, p_2, p_3 and p_4. Here p_1 and p_2 are as above, and p_3 is either *s_known* or *s_unknown* (according to whether s is given to the learning algorithm or not) and p_4 is either *always_halts* or *usually_halts* (according to whether learning algorithms are required to halt or not).

As examples, **functional**($\log(1/\delta)$, *rand*) is the set of all pairs of concept classes (**C**, **H**) such that **C** is polynomially learnable by **H** in the functional model by randomized algorithms with sample complexity polynomially dependent on $\log(1/\delta)$, and **two-oracle**($1/\delta$, *det*, *s_unknown*, *usually_halts*) is the set of all pairs of concept classes (**C**, **H**) such that **C** is polynomially learnable by **H** in the two-oracle model by deterministic algorithms with running time polynomial in $1/\delta$ and no explicit knowledge of s that halt probabilistically.

In the following section we demonstrate that all three of the basic models are equivalent to each other, and that models resulting from all combinations of the above variations are equivalent to each other, except for the restriction mentioned above that if an oracle algorithm does not know s then the halting criterion of the learning algorithm is probabilistic.

In all the models described above the algorithm has to perform well for all probability distributions. For each model one can define learnability with respect to a fixed distribution (or two fixed distributions in the two-oracle case) (Benedek and Itai, 1988a). The question arises which of the equivalences proven in this paper still hold if learnability is defined with respect to fixed distributions. Interestingly enough all equivalences hold in that case as well, with the exception of the equivalence between the one- and two-oracle models.

We make some general assumptions about the concept classes **C** and **H** that hold throughout the following analysis. First we assume that the language used for representing hypotheses in **H** is such that one can efficiently determine if an instance is a member of a given hypothesis. Formally, we assume that there is a polynomial algorithm that, given a string w in L_n and a representation of an instance $x \in X_n$, determines whether or not $x \in \sigma_n(w)$. Such an **H** is called *polynomially evaluatable*. Second, when the domain is real-valued we also assume that all concepts are Borel sets, and that all sets of concepts are well-behaved in the measure-theoretic sense defined in Blumer *et al.* (1989). We let **regular**$_1$ denote the set of all pairs (**C**, **H**) that satisfy the above regularity assumptions.

For certain of the relationships among the models we require stronger regularity assumptions. We let **regular**$_2$ be the set of all pairs (**C**, **H**) in **regular**$_1$ such that for each $H_n \in \mathbf{H}$ we have $\varnothing \in H_n$, $X_n \in H_n$, and for all $x \in X_n$, $\{x\} \in H_n$.

3. *Equivalence Results of Learnability Models*

In this section we prove our main theorem which shows the equivalence of the learnability models introduced in the previous section. As the only restriction we require that if *s unknown*, then the halting criterion must be *usually_halts*.

THEOREM 3.1. *If* $(\mathbf{C}, \mathbf{H}) \in \mathbf{regular}_2$ *is an element of any of the following sets, then it is an element of all of them*:

$$\mathbf{functional}(p_1, p_2),$$
$$\mathbf{one\text{-}oracle}(p_1, p_2, p_3, p_4),$$
$$\mathbf{two\text{-}oracle}(p_1, p_2, p_3, p_4).$$

Here

$$p_1 \in \{\log 1/\delta,\ 1/\delta,\ \textit{fixed}\ \delta\},$$
$$p_2 \in \{\textit{rand},\ \textit{det}\},$$
$$p_3 \in \{s\_\textit{known},\ s\_\textit{unknown}\},$$
$$\textit{if}\ p_3 = s\_\textit{known}\ \textit{then}\ p_4 \in \{\textit{always\_halts},\ \textit{usually\_halts}\},$$
$$\textit{if}\ p_3 = s\_\textit{unknown}\ \textit{then}\ p_4 = \textit{usually\_halts}.$$

Figure 1 presents a graph whose vertices represent the models that we show to be equivalent in this theorem. The directed edges represent the implications that we will directly demonstrate in our proof of the theorem. Some of these edges are labeled with the numbers of the lemmas in which the corresponding implications are demonstrated. The other implications are considered below.

Proof of Theorem 3.1. The implications corresponding to the unlabeled edges of Fig. 1 follow immediately from the following observations. A deterministic algorithm for learning **C** by **H** in some model is also a randomized algorithm for learning **C** by **H** in the model that differs only in replacing the parameter *det* with *rand*. Similarly, a deterministically halting algorithm is also a probabilistically halting algorithm. A learning algorithm for a model in which s is not available to the algorithm can also be used when s is available: it just ignores s. The $1/\delta$ models have been omitted from the diagram, but would fit in the middle of each of the downward pointing arrows between the log $1/\delta$ and fixed-δ models. For any particular set of values of the other parameters, learnability in the log $1/\delta$ model implies learnability in the corresponding $1/\delta$ model, and learnability in the $1/\delta$ model implies learnability in the corresponding fixed-δ model. To see this note that an algorithm that is polynomial in log $1/\delta$ is polynomial in $1/\delta$. An algorithm in either of the oracle models that is polynomial in $1/\delta$