Machine Learning 4, 41–65, 1989

Semi-Supervised Learning

RAYMOND BOARD (board@cs.uiuc.edu)
LEONARD PITT (pitt@cs.uiuc.edu)
Department of Computer Science, University of Illinois at Urbana-Champaign, 1304 W. Springfield Ave., Urbana, IL 61801. USA

Editor: David Haussler

Keywords: concept learning, classification, pac-learning, Boolean formulas, polynomial-time identification.

Abstract. The distribution-independent model of (supervised) concept learning due to Valiant (1984) is extended to that of *semi-supervised learning* (ss-learning), in which a collection of disjoint concepts is to be simultaneously learned with only partial information concerning concept membership available to the learning algorithm. It is shown that many learnable concept classes are also ss-learnable. A new technique of learning, using an *intermediate oracle*, is introduced. Sufficient conditions for a collection of concept classes to be ss-learnable are given.

1. Introduction

Theoretical results in concept learning have received much attention recently within the model of learnability introduced by Valiant (1984). This model (called *pac-learnability*, for "probably approximately correct learning" (Angluin, 1988a)) assumes that there is a teacher, or oracle, that presents the learner with randomly generated examples (and counterexamples) of the concept to be learned. The effects of providing additional information to the learner, such as queries and hints, have also been studied (Valiant, 1984; Angluin, 1987, 1988a, 1988b, 1988c, 1988d; Berman & Roos, 1987).

In this paper we ask whether it is possible to learn with *less* information—without a teacher labeling examples of each concept to be learned as positive or negative. Further, we consider the problem of simultaneously learning a collection of concepts, instead of just a single one.

There are (at least) two situations that we might wish to model that involve learning in an environment with no teacher and many concepts to be learned. One is to assume there are no *a priori* underlying concepts against which the learner is to be evaluated, and that the goal is to partition the examples in a manner consistent with some predetermined criterion. This approach is traditionally known as clustering, or *unsupervised* learning, and has been studied extensively. Summaries may be found in Anderberg (1973), Duda and Hart (1973), Hartigan (1975), Carbonell, Michalski, and Mitchell (1983); Romesburg, (1984).

The other approach, which is undertaken in this paper, is to assume that there are in fact specific concepts to be learned, yet there is no teacher labeling each

element as to its concept membership. In this case, the criterion of success is how well the learned concepts approximate the correct underlying concepts. Of course, in the absence of *any* information about the underlying concepts, and without a predetermined criterion for measuring the suitability of a clustering, the learning task is impossible. If, on the other hand, there is a teacher who labels each element with its corresponding concept name, then (for any reasonable definition of concept learning) the simultaneous (*supervised*) learning of a disjoint collection of concepts trivially reduces to separate instances of learning individual concepts. (For each concept, the positive examples will be the members of the concept, and the negative examples will be members of the other concepts.)

We strike a compromise between these two extremes and investigate the simultaneous learnability of a collection of concepts in a *semi-supervised* manner, i.e., with partial information. Rather than assuming that concept labels are given, we assume instead that there is an oracle that, upon request, will randomly and independently choose two examples from an unknown distribution on the space of possible points and tell the learner whether or not the points belong to the same concept. A possible interpretation or justification of such an oracle is a learning environment in which the learner is able to occasionally and randomly notice that two examples ought to be classified together (or apart), yet does not necessarily have the ability to relate these two examples to other examples previously seen or likely to be seen.

If there is only one concept to be learned, then the problem is closely related to a form of concept learning in which the teacher, rather than providing randomly chosen positive and negative examples, instead answers whether two randomly chosen examples are of the same type, i.e., both positive or both negative, without telling which is the case. Thus the learnability of a single concept in a semi-supervised manner is an interesting question itself, as it explores the boundary of the amount of information that is necessary for concept learning. It would seem that if, in addition, the examples were from many different concepts to be learned simultaneously in a semi-supervised manner, then the learning problem would be significantly more difficult.

We show that, in fact, for a wide range of families F of Boolean formulas known to be pac-learnable, and for every constant $r > 0$, any collection of r disjoint concepts defined by formulas of F is learnable in a semi-supervised manner (ss-learnable) in polynomial time.

Sufficient conditions are given for the ss-learnability of a collection of concepts, where each concept is from some fixed concept class of finite "Vapnik-Chervonenkis dimension" (VC-dimension), a combinatorial parameter of a concept class that is intimately related to the pac-learnability of that class (Blumer, Ehrenfeucht, Haussler, & Warmuth, 1986, 1987). In particular, it is shown that if C has finite VC-dimension, and C is learnable from positive examples only, then any collection of r disjoint concepts from C can be ss-learned in time polynominal in r.

Of particular interest is a new technique of learning an *intermediate oracle*. Many concept classes would be ss-learnable if we were to assume the existence of an

oracle that, when asked about two examples, tells us whether or not they are examples of the same concept. We do not, however, wish to assume the availability of such an oracle. Since we have access to pairs of points labeled as to whether or not they are in the same concept, in many cases we can use these examples to learn a concept description that will imitate the desired oracle quite accurately. We call this concept description an intermediate oracle. Once learned, the intermediate oracle can be used in place of a real oracle. We expect that this technique will prove useful for other learning problems.

The rest of the paper is organized as follows. In Section 2 we review the necessary background and define ss-learnability. Section 3 gives an algorithm for polynomial time ss-learning of monomial concept classes. Section 4 gives sufficient conditions for ss-learnability, which, in Section 5, are used to prove the ss-learnability of other classes of Boolean formulas. In Section 6, the ss-learnability of a concept class is related to the VC dimension of the class, and additional sufficient conditions are given. In Section 7, an ostensibly different definition of ss-learning is given, and shown to be equivalent to that of ss-learnability. Finally, Section 8 summarizes the results of the paper and suggests some directions for further study.

2. Notation, definitions, and background

For each $n \geq 1$, let $X_n = \{x_1, x_2, \ldots, x_n\}$ be a set of n Boolean variables. A family of Boolean formulas is a set $F = \cup_{n \in \mathbf{N}} F_n$, where for each n, F_n is any set of formulas over the variables X_n. For each $f \in F_n$, f defines the *concept* $\{x \in \{0, 1\}^n : f(x) = 1\}$. We will sometimes use f to denote the concept that the formula f defines; thus $x \in f$ means that $f(x) = 1$. Similarly, we will refer to the parameterized family of formulas F as a concept class.[1] An *example* is a element of $\{0, 1\}^n$, where n is given by context. An element $x \in \{0, 1\}^n$ is a positive example (of f) if $x \in f$, and a negative example otherwise. The symbol $\oplus$ denotes the symmetric difference of two sets; thus for $f, f' \in F_n$, $f \oplus f' = \{x : f(x) \neq f'(x)\}$.

If $f \in F_n$ and D is a probability measure defined on elements of $\{0, 1\}^n$ then let $\text{EXAMPLES}_{f,D}$ be an oracle which, when called, randomly chooses an element $x \in \{0, 1\}^n$ according to the distribution D, and reports $(x, +)$ if x is a positive example, and $(x, -)$ if x is a negative example of f. (We call $(x, +)$ and $(x, -)$ *labeled examples*.) If $S \subseteq \{0, 1\}^n$ then let $D(S) = \Sigma_{s \in S} D(s)$.

The definition of pac-learnability of a family of formulas involves two parameters: an *accuracy* parameter ϵ, and a *confidence* parameter δ. Intuitively, a concept class is pac-learnable if there is a polynomial time algorithm that, given access to independently generated labeled examples of some unknown target concept drawn according to any fixed but unknown distribution will, with high confidence (probability at least $1 - \delta$), produce a concept from the class that has high accuracy (i.e., the probability that a randomly generated example is classified differently by the concept produced and the target concept is at most ϵ). Further intuitions and justifications of this model, as well as relationships with previous work in machine

learning, have been discussed by Valiant (1984), Kearns, Li, Pitt, and Valiant (1987b), and Haussler (1988).

Definition 2.1. The family $F = \cup_{n\in\mathbf{N}} F_n$ of Boolean formulas is *pac-learnable* iff there exists an algorithm A and polynomials p and q such that for all $n \geq 1$, for all $f \in F_n$, for every probability distribution D on $\{0, 1\}^n$, and for all $\epsilon, \delta > 0$, if A is given as input $p\left(n, \frac{1}{\epsilon}, \frac{1}{\delta}\right)$ labeled examples generated by $\text{EXAMPLES}_{f,D}$, and the parameters ϵ and δ, then in time $q\left(n, \frac{1}{\epsilon}, \frac{1}{\delta}\right)$ A outputs a formula $f' \in F_n$ such that with probability at least $1 - \delta$, $D(f \oplus f') \leq \epsilon$.

If $D(f \oplus f') \leq \epsilon$, then we say that f' is an ϵ-*approximation* of f (with respect to D), or is ϵ-*accurate* for f (with respect to D), omitting the parenthesized phrase whenever D is clear from context. Thus pac-learnability requires that a learning algorithm exists that, with high probability $(1 - \delta)$, can produce an ϵ-approximation of any unknown target concept from the class of formulas being learned. Further, the time and number of examples used by the learning algorithm may increase at most polynomially in the inverse of the parameters ϵ and δ, and polynomially in the length n of each example.

There are a number of alternate definitions of pac-learnability that have been developed in order to model various additional aspects of the learning task. Our results hold with only minor modifications for most of these definitions of learnability (see, for example, Haussler, Kearns, Littlestone, & Warmuth, 1988).

We review the definitions of some pac-learnable families of Boolean formulas. A literal (of the variable set X_n) is either the symbol x_i or its negation $\bar{x}_i$ for some $i \leq n$. In the following, let k be any fixed natural number.

monomials: $\cup_{n\in\mathbf{N}}\{m : m$ is a conjunct of a subset of $\{x_1, x_2, \ldots, x_n, \bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n\}\}$. The size of a monomial is the number of literals it contains.

***k*DNF:** k-disjunctive normal form formulas $= \cup_{n\in\mathbf{N}}\{f : f$ is a disjunct of monomials, each of size at most k, over n variables$\}$.

***k*CNF:** k-conjunctive normal form formulas $= \cup_{n\in\mathbf{N}}\{f : f$ is a conjunct of clauses, each of size at most k, over n variables$\}$, where a clause is a disjunct of literals and the size of a clause is the number of literals it contains.

***k*-term-DNF:** $\cup_{n\in\mathbf{N}}\{f : f$ is a disjunct of at most k monomials over n variables$\}$.

***k*-clause-CNF:** $\cup_{n\in\mathbf{N}}\{f : f$ is a conjunct of at most k clauses over n variables$\}$.

***k*-decision-lists:** A k-Decision List (over n variables, for any $n \in \mathbf{N}$) is a list of pairs $C = ((m_1, b_1), \cdots, (m_j, b_j))$ where each m_i is a monomial (over n variables) of size at most k and each b_i is 0 or 1. The value of C on $x \in \{0, 1\}^n$ is defined algorithmically: let i be the least number such that x satisfies m_i. Then $C(x) = b_i$ (or 0 if no such i exists).

A variant of pac-learnability (Pitt & Valiant, 1988; Blumer et al., 1987) is the notion of F pac-learnable *in terms of* G (also written F pac-learnable "by" G), where G is some other family of formulas. The definition is the same as pac-learnability, except that the ϵ-approximate formula f' output by the algorithm must be a member of the family G instead of F. As we shall see, this notion is particularly relevant to this work.

Theorem 2.2.

1. Monomials are pac-learnable.
2. For each $k \geq 1$, *kDNF* is pac-learnable.
3. For each $k \geq 1$, *kCNF* is pac-learnable.
4. For each $k \geq 1$, *k-decision-lists* are pac-learnable.
5. For each $k \geq 1$, *k-term-DNF* is pac-learnable in terms of *kCNF*.
6. For each $k \geq 1$, *k-clause-CNF* is pac-learnable in terms of *kDNF*.

1 and 3 are from Valiant (1984), 2 is from Valiant (1985), 4 from Rivest (1987), and 5 and 6 from Pitt and Valiant (1988). See also papers by Blumer et al. (1987), Kearns et al. (1987a, 1987b), Haussler (1988), and Littlestone (1988).

A relaxation of the definitions of pac-learnable to that of *prediction* has also been studied (Littlestone, 1988; Haussler, Littlestone, & Warmuth, 1988; Pitt & Warmuth, 1988; Haussler, Kearns, Littlestone, & Warmuth, 1988). We say that a class of concepts is (polynomially) predictable if Definition 2.1 holds, except instead of being required to output a formula $f' \in F_n$ that is an ϵ-approximation of the target concept f, the learning algorithm A may output *any polynomial time algorithm* M such that the function computed by M is an ϵ-approximation of f. Clearly each of the classes mentioned in Theorem 2.2 is predictable.[2]

We now naturally extend the definition of pac-learning to the ss-learning of r disjoint concepts. Let $r \in \mathbf{N}$. Let $F = \cup_{n \in \mathbf{N}} F_n$ be a family of formulas, and for some n, let $f_1, f_2, \ldots, f_r \in F_n$ be pairwise disjoint (i.e., the sets of satisfying assignments of the f_i's are disjoint). Let D be any probability distribution on $\{0, 1\}^n$ such that

$$D(\cup_{i=1}^{r} f_i) = 1. \qquad [1]$$

Thus the only elements that may occur when sampling from D are those which satisfy one (and hence exactly one) of the f_i's.

Let LABELED-PAIRS$_{f_1, f_2, \ldots, f_r, D}$ be an oracle that, when called, randomly and independently chooses two elements $x, y \in \{0, 1\}^n$ according to the distribution D, and returns (x, y, *same*) if for some i, x and y both satisfy f_i, or returns (x, y, *different*) if x and y satisfy different elements of $\{f_1, f_2, \ldots, f_r\}$. When D and f_1, $f_2, \ldots, f_r$ are clear from context, we omit the subscripts and write only LABELED-PAIRS.

An alternative definition would permit D to generate examples that are not in any of the f_i's. However, it is then more difficult to find a natural definition of LABELED-PAIRS. (It is not clear how a pair (x, y) should be labeled if one or both elements are not in any concept f_i.)

To define the learning of a collection of r concepts $\mathcal{F} = \{f_1, f_2, \ldots, f_r\}$ in a semi-supervised manner, we need to measure the error of a collection of s concepts $\mathcal{G} = \{g_1, g_2, \ldots, g_s\}$ with respect to $\mathcal{F}$ and a given distribution D. The definition is obtained by the following intuitive considerations: Ideally, $s = r$ and there is a correspondence between elements of $\mathcal{G}$ and $\mathcal{F}$ such that each g_i is an approximation of some unique concept f_j. However, it is conceivable that more or fewer than the true number of concepts were learned, and thus there may be no correspondence between some of the elements of $\mathcal{G}$ and some of the elements of $\mathcal{F}$. Let $\mathcal{G}' \subseteq \mathcal{G}$ be the set of those elements of $\mathcal{G}$ for which there is in fact a corresponding element of $\mathcal{F}$. We measure the error of $\mathcal{G}$ in the following way: An element $x \in \{0, 1\}^n$ is an error point if any of the following conditions hold.

Error-1 x is not in any given $g_i \in \mathcal{G}'$. The intention is that $\mathcal{G}'$ contains the relevant learned concepts—those corresponding to the underlying concepts of $\mathcal{F}$. Thus any point falling outside of the region $\cup\mathcal{G}'$ should be counted towards the error.

Error-2 x is in the symmetric difference of some $g_i \in \mathcal{G}'$ with the corresponding f_j. This counts as error any discrepancy between a learned concept and the corresponding underlying concept that it is intended to approximate.

Error-3 x is in the intersection of two different concepts in $\mathcal{G}$. This prohibits excessive overlap among the learned concepts.

To formally restate the above regions of error, let $I : \mathcal{G}' \to \mathcal{F}$ be an injection mapping learned concepts of $\mathcal{G}'$ to their corresponding underlying target concepts of $\mathcal{F}$. Then

- $E1 = \overline{\cup\mathcal{G}'}$
- $E2 = \cup_{g_i \in \mathcal{G}'}(g_i \oplus I(g_i))$
- $E3 = \cup_{g_i, g_j \in \mathcal{G}, i \neq j}(g_i \cap g_j)$

Now we may define how closely a finite collection $\mathcal{G}$ of concepts approximates a finite collection $\mathcal{F}$ of concepts.

Definition 2.3. Given $\mathcal{F} = \{f_1, f_2, \ldots, f_r\}$, $\mathcal{G} = \{g_1, g_2, \ldots, g_s\}$, and a distribution D on $\{0, 1\}^n$ satisfying equation (1), then $\mathcal{G}$ is ϵ-*close* to $\mathcal{F}$ iff there exists a subset $\mathcal{G}' \subseteq \mathcal{G}$ and an injection $I : \mathcal{G}' \to \mathcal{F}$ such that

$$D(E1 \cup E2 \cup E3) \leq \epsilon.$$

The motivation for counting the regions $E1$ and $E2$ as error associated with $\mathcal{G}$ should be clear. The purpose of region $E3$ is to preclude the possibility of making

$\mathcal{G}$ so large that the difficulty in the learning problem becomes determining the subset $\mathcal{G}'$ that has the desired properties. For example, if the third error component was omitted, any family F of formulas could be ss-learned (as defined below) by simply setting $\mathcal{G} = 2^{\{0,1\}^n}$ so that $\mathcal{G}$ contains every possible concept over n variables. Limiting the amount of overlap among the concepts of $\mathcal{G}$ appears to be the best among a number of different ways around this problem.

Finally, the following definition of ss-learnability parallels that of pac-learnability, except that the information available to an ss-learning algorithm consists of LABELED-PAIRS, and the algorithm is required to produce a collection of concepts that is ϵ-close to the underlying collection of concepts.

Definition 2.4. The family $F = \cup_{n \in \mathbb{N}} F_n$ of Boolean formulas is *ss-learnable* (learnable in a semi-supervised manner) iff for each number $r > 0$, there exists an algorithm A and polynomials p and q such that for all $n \geq 1$, for every disjoint collection $\{f_1, f_2, \ldots, f_r\} \subseteq F_n$, for any probability distribution D on $\{0, 1\}^n$ satisfying equation (1), and for all $\epsilon, \delta > 0$, if A is given as input the parameters ϵ and δ, and at most $p\left(n, \frac{1}{\epsilon}, \frac{1}{\delta}\right)$ labeled pairs generated by LABELED-PAIRS$_{f_1, f_2, \ldots, f_r, D}$, then in time $q\left(n, \frac{1}{\epsilon}, \frac{1}{\delta}\right)$, A outputs a collection $\{g_1, g_2, \ldots, g_s\} \subseteq F_n$ that, with probability at least $1 - \delta$, is ϵ-close to $\{f_1, f_2, \ldots, f_r\}$.

Note that this definition does not require that the learned concepts be disjoint, even though the concepts which they approximate are disjoint. However, any area of overlap among the learned concepts would contribute towards the allowable error (region $E3$). Similarly, the union of the learned concepts is not required to exhaust the instance space, but any region that is in one of the original concepts but in none of the learned concepts is counted towards the error (region $E1$).

Note also that we allow the algorithm to depend on the number of concepts r to be learned. In particular, the run time need not be polynomial in r. Of course, it would be preferable to have an algorithm that always runs in time polynomial in r. We have not been able to extend our results in this manner. This is similar to the problem of pac-learning, where for many concept classes (e.g. DNF), although it is not known whether the class as a whole is pac-learnable, positive learnability results have been found for subclasses in which some measure of the concept size is assumed to be bounded by a constant (e.g., k-DNF).

3. Semi-supervised learning of monomials

In this section we assume that the concepts to be learned are monomials m_1, $m_2, \ldots, m_r$ over n variables. We show that, given access to randomly generated pairs of strings from concepts defined by monomials, labeled only as to whether or not they are members of the same concept, we can learn monomials that accurately describe the concepts in polynomial time. In Sections 4 and 5 we generalize the techniques to ss-learn other collections of concepts.

Theorem 3.1. Monomials are ss-learnable.

The general idea of the proof of Theorem 3.1 is as follows. Suppose we have an oracle that can tell us for any pair of vectors x and y whether or not they are in the same concept, i.e., satisfy the same monomial. Then we can learn a set G of monomials that satisfies the definition of ss-learnability. This can be done by collecting enough individual points, say m of them (obtained by $m/2$ calls to LABELED-PAIRS), to ensure that, with high probability, we have at least one representative from each concept of significant weight with respect to the distribution. Then, using our assumed oracle, we can query each of the $\binom{m}{2}$ pairs of these points to see which of them are in the same concept, and use the results to build equivalence classes. We can then use the m points as examples (positive or negative, depending on the particular concept) with which to learn the monomials defining membership in the particular concepts. Thus m must also be large enough so that, with high probability, the learned monomials are sufficiently accurate.

We don't have such an oracle; what we *do* have is LABELED-PAIRS, which can give us *same* and *different* labels, but only for randomly generated pairs of points, not for requested pairs. We cannot just wait for the pairs that we are interested in to be generated, since D may be such that this would take exponential time. What we will do to get around this problem is to *learn* an oracle from the examples of sameness and differentness supplied by LABELED-PAIRS. Again, it might take too long to learn an oracle that responds correctly on all possible inputs; we instead learn an *approximate* oracle, and guarantee that the approximate oracle is accurate enough so that, with high probability, it will be correct on each pair of points in our sample of size m. (We call such an oracle an *intermediate oracle,* as it is not supplied to the learning algorithm, but is constructed and used by the learning algorithm as an oracle enabling a solution of the proper form to be discovered.)

Definition 3.2. For any collection of monomials $m_1, \ldots, m_r$ over the variable set X_n, the concept SameConcept $(\langle m_1, \ldots, m_r \rangle) \subseteq \{0, 1\}^{2n}$ is the set of all vectors of length $2n$ such that the first n-bit substring and the second n-bit substring are in the same monomial concept; i.e., if x and y are n-bit strings and their concatenation $xy \in$ SameConcept $(\langle m_1, \ldots, m_r \rangle)$, then for some i, $1 \leq i \leq r$, $x \in m_i$ and $y \in m_i$. When clear from context, we omit the argument $\langle m_1, \ldots, m_r \rangle$ and refer to the concept as SameConcept.

Note that the oracle LABELED-PAIRS$_{m_1, m_2 \ldots, m_r, D}$ is a generator of positive and negative examples for the concept SameConcept $(\langle m_1, \ldots, m_r \rangle)$ (with product distribution $D^2 : X_{2n} \to \mathrm{R}$ defined by $D^2(z) = D(x)D(y)$, where $z = xy$).

Lemma 3.3. The class of concepts $\cup_n \{\text{SameConcept } (\langle m_1, \ldots, m_r \rangle) : m_1, \ldots, m_r$ are monomials over the variable set $X_n\}$ is pac-learnable in terms of rCNF.

Proof. Let x and y be n-bit strings, and let z be their concatenation, xy. Suppose

that $z \in \text{SameConcept}(\langle m_1, \ldots, m_r \rangle)$. Then $x, y \in m_i$ for some $i \leq r$. For each $i \leq r$, let the monomial (over $2n$ bits) m_i' be defined such that for all $j \leq n$,

$$(x_j \in m_i') \wedge (x_{n+j} \in m_i') \Leftrightarrow x_j \in m_i$$

and

$$(\bar{x} \in m_i') \wedge (\bar{x}_{n+j} \in m_i') \Leftrightarrow \bar{x}_j \in m_i.$$

Thus z must satisfy m_i'. In fact, the strings that satisfy m_i' are exactly those strings that are concatenations of pairs of strings that both satisfy m_i. Thus the set of pairs of strings that are in the same concept is the set of strings whose concatenations satisfy the r-term DNF expression $m_1' \vee m_2' \vee \ldots \vee m_r'$. This means that the pairs of points that are in the same (different) concept are the pairs whose concatenations are positive (negative) examples of an r-term DNF expression over $2n$ variables. Although for each $r \geq 2$ it is NP-hard to learn an ϵ-accurate r-term DNF expression from examples (Pitt & Valiant, 1988), r-term DNF *is* pac-learnable in terms of rCNF (Theorem 2.2). Thus we can obtain (with probability at least $1 - \delta$) an rCNF expression that is ϵ-accurate for $m_1' \vee \ldots \vee m_r'$ in time polynomial in n, $\frac{1}{\epsilon}$, and $\frac{1}{\delta}$. The rCNF expression will be used as an oracle SC for the concept SameConcept in the learning algorithm sketched below. □

Definition 3.4. For any p, $0 < p < 1$, a *p-significant concept* $m_i \in \{m_1, m_2, \ldots, m_r\}$ is one for which $D(m_i) \geq p$. Elements of $\{m_1, m_2, \ldots, m_r\}$ that are not p-significant are *p-insignificant*.

Note that

$$D\left(\bigcup_{m_i \text{ is } \epsilon/2r \text{ -insignificant}} m_i\right) \leq \sum_{m_i \text{ is } \epsilon/2r \text{ -insignificant}} D(m_i) \leq \sum_{i=1}^{r} \frac{\epsilon}{2r} = \frac{\epsilon}{2}. \qquad [2]$$

We now sketch the ss-learning algorithm. Let m be a number to be specified later. The size of m will be at most polynomial in n, $\frac{1}{\epsilon}$, and $\frac{1}{\delta}$.

Monomial ss-Learning Algorithm

1. Use LABELED-PAIRS to learn, with probability at least $1 - \frac{\delta}{4}$, an oracle SC for SameConcept that is $\frac{\delta}{2m^2}$-accurate. A learning algorithm exists by Lemma 3.3. The time taken (and hence the number of examples needed) is at most $p_1\left(2n, \frac{2m^2}{\delta}, \frac{4}{\delta}\right)$, where $p_1\left(n, \frac{1}{\epsilon}, \frac{1}{\delta}\right)$ is the (polynomial) time needed to learn an rCNF expression of n variables with accuracy ϵ and confidence δ.
2. Make $\frac{m}{2}$ calls to LABELED-PAIRS to obtain, with probability at least $1 - \frac{\delta}{4}$,

an m-sample (a sample of size m) containing at least one element from each $\frac{\epsilon}{2r}$-significant monomial.
3. Use SC to divide the m-sample into equivalence classes in the obvious way.
4. For each equivalence class, label the m-sample accordingly and input the sample, together with accuracy and confidence parameters $\frac{\epsilon}{2r}$ and $\frac{\delta}{4r}$ respectively, to any algorithm for pac-learning of monomials. Each monomial output by the learning algorithm is output as a concept description.

Theorem 3.5. Let the variable m in the Monomial ss-Learning Algorithm be such that $m = max\left\{\frac{2r}{\epsilon}\ln\frac{4r}{\delta}, p_2\left(n, \frac{2r}{\epsilon}, \frac{4r}{\delta}\right)\right\}$, where p_2 is the (polynomial) number of examples needed by to pac-learn a monomial of n variables with accuracy parameter $\frac{\epsilon}{2r}$ and confidence parameter $\frac{\delta}{4r}$. Then the Monomial ss-Learning Algorithm ss-learns r disjoint monomial concepts.

Clearly Theorem 3.1 follows from Theorem 3.5. To prove Theorem 3.5 we need the following definition and lemmas.

Definition 3.6. A *good run* of the Monomial ss-Learning Algorithm is one in which

(A) The oracle SC learned in step 1 is in fact a $\frac{\delta}{2m^2}$-approximation of SameConcept;
(B) The m-sample obtained in step 2 does have at least one element of every $\frac{\epsilon}{2r}$-significant monomial in $\{m_1, m_2, \ldots, m_r\}$;
(C) In step 3, SC makes no mistakes in dividing the particular m-sample into equivalence classes; and
(D) Each learned monomial concept from step 4 of the algorithm is in fact an $\frac{\epsilon}{2r}$-approximation of the (real) monomial that covers the elements of the equivalence class labeled as positive examples.

Lemma 3.7. Let $\mathcal{M}$ be the set of monomials to be learned. If a good run of the Monomial ss-Learning Algorithm occurs, and $\mathcal{G}$ is the set of monomials produced, then $\mathcal{G}$ is ϵ-close to $\mathcal{M}$.

Proof. Set $\mathcal{G}'$ (as described in Definition 2.3) equal to $\mathcal{G}$. Let I be the injection mapping each learned monomial $g \in \mathcal{G}'$ to the (real) monomial of $\mathcal{M}$ that is consistent with the labeling of the m-sample that was used to learn g. (By (C), there is exactly one such monomial.) Since $\mathcal{G}' = \mathcal{G}$, and since each x such that $D(x) > 0$ is in exactly one $m_i \in \{m_1, \ldots, m_r\}$, then for any i, j such that g_i and g_j are in $\mathcal{G}'$ and $i \neq j$,

$$x \in g_i \cap g_j \Rightarrow x \in g_i \oplus I(g_i) \vee x \in g_j \oplus I(g_j).$$

Thus

$$E3 = \bigcup_{g_i, g_j \in \mathcal{G}, i \neq j} (g_i \cup g_j) \subseteq \bigcup_{g_i \in \mathcal{G}'} (g_i \oplus I(g_i)) = E2,$$

so to show that $\mathcal{G}'$ is ϵ-close to $\mathcal{M}$ it suffices to show that

$$D(E1 \cup E2) \leq \epsilon.$$

By (D), for each $g \in \mathcal{G}'$, $g \oplus I(g) \leq \frac{\epsilon}{2r}$, and since there are at most r elements of $\mathcal{G}'$,

$$D(E2) = D\left(\bigcup_{g_i \in \mathcal{G}'} g_i \oplus I(g_i) \right) \leq r \frac{\epsilon}{2r} = \frac{\epsilon}{2}.$$

To show that $\mathcal{G}'$ is ϵ-close to $\mathcal{M}$, it thus suffices to show that

$$D(E1 - E2) \leq \frac{\epsilon}{2}. \qquad [3]$$

By equation (2), equation (3) is true if

$$E1 - E2 \subseteq \cup\{m_i : m_i \text{ is } \epsilon/2r\text{-insignificant}\}, \qquad [4]$$

which is true if

$$E1 \cap \cup\{m_i : m_i \text{ is } \epsilon/2r\text{-significant}\} \subseteq E2. \qquad [5]$$

To see that containment (5) holds, note that if $x \in m_i$ and m_i is $\frac{\epsilon}{2r}$-significant, then by (B), there is some $g \in \mathcal{G}'$ such that $I(g) = m_i$. If x is also in $E1 = \overline{\cup \mathcal{G}'}$, then $x \notin g$, so $x \in g \oplus I(g) \subseteq E2$. □

Lemma 3.8. Let SC be a $\frac{\delta}{2m^2}$-approximation of SameConcept (with respect to the product measure D^2). If we randomly generate m points (from $\frac{m}{2}$ calls to LABELED-PAIRS), then with probability at least $1 - \frac{\delta}{4}$, SC will correctly classify each of the $\binom{m}{2}$ pairs of points as to sameness and differentness.

Proof. Let SC be a $\frac{\delta}{2m^2}$-approximation of SameConcept, and thus $\frac{\delta}{2m^2}$-accurate with respect to the oracle LABELED-PAIRS. Consider any m-sample generated randomly from $\frac{m}{2}$ calls to LABELED-PAIRS. Number all of the pairs of points in the m-sample from 1 to $\binom{m}{2}$. For each i from 1 to $\binom{m}{2}$, let γ_i be the probability that the oracle SC is wrong on the i^{th} of the $\binom{m}{2}$ pairs. Since, for any m-sample, each

permutation of the m points is equally likely, $\left(\forall i, j \leq \binom{m}{2}\right) \gamma_i = \gamma_j$. In particular, $\left(\forall i \leq \binom{m}{2}\right) \gamma_i = \gamma_1$. Note that γ_1 is the probability that a $\frac{\delta}{2m^2}$-accurate oracle SC is wrong on the first pair. Thus the probability that SC is wrong on *some* pair is at most

$$\sum_{i=1}^{\binom{m}{2}} \gamma_i = \binom{m}{2} \gamma_i \leq \binom{m}{2} \frac{\delta}{2m^2} < \frac{\delta}{4}.$$

□

Lemma 3.9. If m is as in the hypothesis of Theorem 3.5 then the probability of a good run of the Monomial ss-Learning Algorithm is at least $1 - \delta$.

Proof. Let A, B, C, and D represent the events that (A), (B), (C), and (D) (as given in Definition 3.6) occur respectively. The probability of not obtaining a good run is then at most

$$\Pr(\overline{A}) + \Pr(A \cap \overline{B}) + \Pr(A \cap B \cap \overline{C}) + \Pr(A \cap B \cap C \cap \overline{D}).$$

Then

- $\Pr(\overline{A})$ is at most $\frac{\delta}{4}$, by the definition of pac-learnability.
- By hypothesis, $m \geq \frac{2r}{\epsilon} \ln \frac{4r}{\delta}$, thus the probability that B fails to occur, i.e., that some $\frac{\epsilon}{2r}$-significant monomial does not have a representative in the m-sample, is at most

$$r\left(1 - \frac{\epsilon}{2r}\right)^{\frac{2r}{\epsilon} \ln \frac{4r}{\delta}} \leq \frac{\delta}{4}.$$

  Hence $\Pr(A \cap \overline{B}) \leq \Pr(\overline{B}) \leq \frac{\delta}{4}$.
- By Lemma 3.8, the probability that (C) fails to occur given that (A) occurs is at most $\frac{\delta}{4}$, thus $\Pr(A \cap B \cap \overline{C}) \leq \Pr(A \cap \overline{C}) \leq \Pr(\overline{C} \mid A) \leq \frac{\delta}{4}$.
- Given that A, B, and C occurred, the probability that a particular learned monomial has error more than $\frac{\epsilon}{2r}$ is at most $\frac{\delta}{4r}$, by the definition of pac-learnability and the fact that sufficiently many correctly labeled randomly generated examples were input to a monomial learning algorithm. Since there were at most r monomial equivalence classes obtained from the m-sample, the probability that (D) fails to occur, i.e., that some learned monomial has error more than $\frac{\epsilon}{2r}$ is at most $\frac{\delta}{4}$. Thus $\Pr(A \cap B \cap C \cap \overline{D}) \leq \Pr(\overline{D} \mid A \wedge B \wedge C) \leq \frac{\delta}{4}$.

Thus the probability of not obtaining a good run is at most $\frac{\delta}{4} + \frac{\delta}{4} + \frac{\delta}{4} + \frac{\delta}{4} = \delta$. □

To complete the proof of Theorem 3.5 (and hence 3.1), note that by Lemma 3.9, the probability is at least $1 - \delta$ that a good run occurs, implying (by Lemma 3.7) that the set of monomials output by the Monomial ss-Learning Algorithm is ϵ-close to the set of monomials to be learned. □

4. A sufficient condition for ss-learning

In the proof of Lemma 3.3, the pairs of n bit strings that were generated by LABELED-PAIRS were concatenated into a single $2n$ bit string. It was then shown that the concept class corresponding to pairs labeled as "same" was learnable in terms of rCNF. Notice that all in fact that was required was that the concept of sameness was *predictable* (as defined following Theorem 2.2). To apply this technique in general, we will need the following definition.

Definition 4.1. Let $F = \cup_{n \in \mathbf{N}} F_n$ be a family of formulas. Then let

- $FF_{2n} = \{f(x_1, x_2, \ldots, x_n) \wedge f(x_{n+1}, x_{n+2}, \ldots, x_{2n}) : f \in F_n\}$, i.e., the collection of formulas over $2n$ variables obtained by conjoining two copies of any formula f from F_n, such that each variable x_j appearing in the second description is changed to x_{n+j}.
- $\bigvee_r FF_{2n} = \{f_1 \vee f_2 \vee \ldots \vee f_r : f_i \in FF_{2n}, 1 \leq i \leq r\}$.
- $\bigvee_r FF = \cup_{n \in \mathbf{N}} \bigvee_r FF_{2n}$.

Theorem 4.2. If F is pac-learnable and $\bigvee_r FF$ is predictable then F is ss-learnable.

Proof. The proof is a straightforward generalization of the proof of Theorem 3.1. We outline it here. Assume the existence of A, a pac-learning algorithm for F, that uses at most $p_A\left(n, \frac{1}{\epsilon}, \frac{1}{\delta}\right)$ examples, and runs in time bounded by $q_A\left(n, \frac{1}{\epsilon}, \frac{1}{\delta}\right)$ where p_A and q_A are polynomials. We also assume the existence of B, an algorithm that predicts $\bigvee_r FF$, and uses at most $p_B\left(2n, \frac{1}{\epsilon}, \frac{1}{\delta}\right)$ examples, and runs in time at most $q_B\left(2n, \frac{1}{\epsilon}, \frac{1}{\delta}\right)$, where p_B and q_B are polynomials.

To ss-learn r disjoint concepts $f_1, f_2, \ldots, f_r$ of F, examples of SameConcept ($\langle f_1, f_2, \ldots, f_r \rangle$) are constructed by forming the conjuncts of the pairs of strings output by LABELED-PAIRS. The number of such examples constructed is $p_B\left(2n, \frac{2m^2}{\delta}, \frac{4}{\delta}\right)$, which are then input to the algorithm B, along with the parameters n, $\epsilon_B = \frac{\delta}{2m^2}$ and $\delta_B = \frac{\delta}{4}$, where $m = max\left\{\frac{2r}{\epsilon} \ln \frac{4r}{\delta}, p_A\left(n, \frac{2r}{\epsilon}, \frac{4r}{\delta}\right)\right\}$. Then B, in time $q_B\left(2n, \frac{2m^2}{\delta}, \frac{4}{\delta}\right)$, outputs some polynomial time algorithm M that is used as an oracle for the concept SameConcept. A set of m points is then randomly generated, using LABELED-PAIRS, and M is applied to every pair of the m points to obtain equivalence classes. The algorithm A is run once for each of the equivalence classes, using the error parameters $\epsilon_A = \frac{\epsilon}{2r}$ and $\delta_A = \frac{\delta}{4r}$. The time needed is again at most polynomial in

all of the relevant parameters. An analysis identical to the one for the monomial case yields that the concepts learned by A are ϵ-close to the true concepts with probability at least $1 - \delta$. □

5. ss-Learning other Boolean formulas

The sufficient conditions of Theorem 4.2 are applied to show as corollaries that for each k, kDNF, kCNF, and k-decision-lists are ss-learnable.

Corollary 5.1. For any constant k, the family of kCNF formulas is ss-learnable.

Proof. If F is the family of kCNF expressions, then for each n, FF_{2n} is a family of kCNF expressions, since the conjunction of two kCNF expressions is also a kCNF expression. Then $\bigvee_r FF$, the disjunct of r kCNF expressions, may be represented by an rkCNF expression without more than a polynomial increase in size, since r and k are constants. To see this, let the disjunction be

$$\bigvee_{i=1}^{r} E_i$$

where each E_i is a kCNF expression. This is equivalent to the rkCNF expression

$$\bigwedge (c_{j_1} \vee c_{j_2} \vee \cdots \vee c_{j_r})$$

where the conjunction is taken over all possible choices of clauses $c_{j_1} \in E_1$, $c_{j_2} \in E_2$, . . ., $c_{j_r} \in E_r$. Thus we can learn $\bigvee_r FF$ in terms of rkCNF expressions (such expressions are pac-learnable by Theorem 2.2). Consequently, $\bigvee_r FF$ is predictable. Since the family F itself is pac-learnable, the result follows from Theorem 4.2. □

Corollary 5.2. For any constant k, the set of kDNF formulas is ss-learnable.

Proof. Let F be the family of kDNF expressions. For each n, FF_{2n} is a set of $2k$DNF expressions, since the conjunct of two kDNF expressions can be described by a $2k$DNF expression. The family of disjunctions of r such expressions, $\bigvee_r FF$, is also a set of $2k$DNF expressions. Thus $\bigvee_r FF$ can be pac-learned in terms of $2k$DNF expressions using any of the algorithms of Valiant (1985), Blumer et al. (1987), Haussler (1988), or Littlestone (1988). Since $2k$DNF expressions are thus predictable, and F is pac-learnable, the result follows from Theorem 4.2. □

Corollary 5.3. For any constant k, the set of k-decision lists is ss-learnable.

Proof. Let F be the family of k-decision lists. The family FF_{2n} can be described by a $2k$-decision list as follows. The monomials of the new list are formed by

conjoining one monomial from each of the two original lists; they are given a label of 1 if both of the constituent monomials are labeled with 1's; otherwise, they are given a label of 0. The new labeled monomials are then sorted so that

1. Every monomial with first half m_i occurs before every monomial with first half m_j if m_i occurs before m_j in the first decision list.
2. For all monomials with the same first half, every monomial with the second half m_k occurs before every monomial with second half m_l iff m_k occurs before m_l in the second decision list.

The disjunction of two k-decision lists can be represented by a $2k$-decision list which is constructed in a manner similar to the conjunctive case. Thus the disjunction of r $2k$-decision lists can be represented by a $2rk$-decision list, which is pac-learnable (Theorem 2.2). Thus $\vee_r FF$ is pac-learnable in terms of $2rk$-decision lists and is therefore predictable. Since F is pac-learnable, the result follows from Theorem 4.2. □

6. Unparameterized ss-learning and the VC dimension

As seen from Theorem 4.2, in order for our ss-learning algorithm to be applied successfully to a pac-learnable concept class F, it is sufficient that the class $\vee_r FF$ be predictable. In this section we give sufficient conditions for the class $\vee_r FF$ to be predictable, when the concept class F is over an *unparameterized domain* and has *finite VC dimension*, as defined below.

Thus far we have exclusively discussed concept classes F that consisted of Boolean formulas, and hence were defined as infinite collections of subclasses ($F = \cup_{n \geq 0} F_n$) parameterized by n, the number of variables in the formula. For any fixed n, the pac-learnability of any subclass of formulas F_n is uninteresting, because there are at most a finite number of possible formulas, and a naive exhaustive search technique can be shown to successfully pac-learn. However, nontrivial learning problems do arise over a single domain X (as opposed to the parameterized domains $\{0, 1\}^n$), when X is infinite. For example, if X is the Euclidean plane, we may be interested in the learnability of concepts that consist of rectangles with sides parallel to the coordinate axes. We define these problems formally following (Blumer et al., 1986).

If X is any set, then a *concept* c of X is any subset of X. A *concept class* C is any collection of concepts of X. Associated with C is some reasonable encoding scheme for elements of C.

Definition 6.1. A *concept class* C (over domain X) is polynomially learnable iff there exists an algorithm A, and polynomials p and q, such that for all $c \in C$, for every probability distribution D on X, and for all $\epsilon, \delta > 0$, if A is given as input $p\left(\frac{1}{\epsilon}, \frac{1}{\delta}\right)$ labeled examples generated by $\text{EXAMPLES}_{c,D}$, and the parameters ϵ and δ,

then in time $q\left(\frac{1}{\epsilon}, \frac{1}{\delta}\right)$ A outputs the description of a concept $c' \in C$ such that with probability at least $1 - \delta$, $D(c \oplus c') \leq \epsilon$.

Thus polynomial learnability is identical to pac-learnability, except the domain parameter n has been eliminated. Similarly, the definitions of predictability and ss-learnability for unparameterized concept classes exactly follow the definitions of predictability and ss-learnability in the parameterized case, except the parameter n is omitted.

From Theorem 4.2, we saw that (in the parameterized case), the predictability of $\bigvee_r FF$ plays an important role in the application of our general technique. Similarly, for any (unparameterized) concept class C, the predictability of the associated concept class $\bigvee_r CC$ (defined below) will be relevant.

Definition 6.2. Let $C \subseteq 2^X$ be a concept class. Then

- for any $c_1, c_2 \in C$, the concept $c_1 \times c_2$ (over domain $X \times X$) is the concept $\{\langle x, y\rangle : x \in c_1, y \in c_2\}$.
- the concept class $C \times C$ (over $X \times X$) is the set of concepts $\{c_1 \times c_2 : c_1, c_2 \in C\}$.
- the concept class CC (over $X \times X$) is the set of concepts $\{c \times c : c \in C\}$.
- the concept class $\bigvee_r CC$ (over $X \times X$) is the set of concepts $\{c_1 \vee c_2 \vee \cdots \vee c_r : c_i \in CC\}$.

Corollary 6.3. Let $C \subseteq 2^X$ be a concept class such that C is polynomially learnable, and $\bigvee_r CC$ is predictable. Then C is ss-learnable.

Proof. This may be viewed as a special case of Theorem 4.2. □

We will refine the sufficient condition of Corollary 6.3 by incorporating sufficient conditions for the predictability of $\bigvee_r CC$. In order to do this, we will rely on a characterization of the polynomially learnable concept classes due to (Blumer et al., 1987). To state the relevant necessary and sufficient conditions for polynomial learnability, we need a number of preliminary definitions.

Definition 6.4. Given a nonempty concept class $C \subseteq 2^X$ and a set of points $S \subseteq X$, $\Pi_C(S)$ denotes the set of all subsets of S that can be obtained by intersecting S with a concept in C, i.e., $\Pi_C(S) = \{S \cap c : c \in C\}$. If $\Pi_C(S) = 2^S$, then we say that S is shattered by C. The Vapnik-Chervonenkis (VC) dimension of C is the cardinality of the largest finite set of points $S \subseteq X$ that is shattered by C. If arbitrarily large finite sets are shattered, then the VC dimension of C is infinite.

Definition 6.5. If $C \subseteq 2^X$ is a concept class, then a randomized polynomial time hypothesis finder for C is a randomized polynomial time algorithm that takes as input a finite labeled sample of a concept in C, and for some $\gamma > 0$, with probability

at least γ produces (the description of) a concept in C that is consistent with the labeled sample. (A concept c is consistent with a labeled sample if every positive example in the sample is an element of c, and no negative example is an element of c.)

The following theorem is a special case of Theorem 3.1.1 in (Blumer et al., 1987).

Theorem 6.6. If C is a concept class over domain X, then C is polynomially learnable if and only if the VC dimension of C is finite and there is a randomized polynomial time hypothesis finder for C.

The following is a slight variant of Theorem 3.2.4 from Blumer et al. (1987).

Lemma 6.7. If $C \subseteq 2^X$ is a concept class that is polynomially learnable, then $\vee,C$ is polynomially predictable. Further, the time required is polynomial in r as well as $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$.

Proof. Modify the proof of Theorem 3.2.4 of Blumer et al. (1987) in a straightforward manner to allow for a *randomized* polynomial time hypothesis finder, instead of a deterministic one. □

We now prove a sufficient condition for ss-learnability of an (unparameterized) concept class.

Theorem 6.8. If $C \subseteq 2^X$ is a concept class such that C is polynomially learnable and there exists a randomized polynomial time hypothesis finder for CC, then C is ss-learnable.

Proof. By Corollary 6.3, it is sufficient to show that $\vee,CC$ is predictable. Since C is polynomially learnable, by Theorem 6.6, C has finite VC dimension. By Lemma 6.9 below, CC also has finite VC dimension. This, together with the hypothesis that CC has a randomized polynomial time hypothesis finder (and an application of Theorem 6.6 once again), implies that CC is polynomially learnable. Finally, applying Lemma 6.7 with CC in place of C, we conclude that $\vee,CC$ is predictable. □

Lemma 6.9. If C has (finite) VC dimension d, then CC has (finite) VC dimension at most $4d \log 6$.

Proof. For any concept class C, let $\mathrm{VCdim}(C)$ denote the VC dimension of C. Note that $CC \subseteq C \times C$, so clearly $\mathrm{VCdim}(CC) \leq \mathrm{VCdim}(C \times C)$. We show that $\mathrm{VCdim}(C \times C) \leq 4d \log 6$.

Let the concept class $C \times X$ (over domain $X \times X$) be the set $\{c \times X : c \in C\}$. Similarly, define $X \times C = \{X \times c : c \in C\}$. We claim that $\mathrm{VCdim}(C \times X) =$

$\text{VCdim}(X \times C) = \text{VCdim}(C)$. We only show that $\text{VCdim}(C \times X) = \text{VCdim}(C)$. The proof for $X \times C$ is virtually identical.

To see that $\text{VCdim}(C \times X) \geq \text{VCdim}(C)$, note that if $S \subseteq X$ is a set of points that is shattered by C, then $S \times \{x\}$ is shattered by $C \times X$, for any particular point $x \in X$. To show that $\text{VCdim}(C \times X) \leq \text{VCdim}(C)$, let $\{\langle x_1, y_1\rangle, \langle x_2, y_2\rangle, \ldots \langle x_d, y_d\rangle\}$ be shattered by $C \times X$. Observe that for every $c \times X \in C \times X$, and for all $x, y, y' \in X$, we have $\langle x, y\rangle \in c \times X$ if and only if $\langle x, y'\rangle \in c \times X$. Thus we can replace $y_1, y_2, \ldots y_d$ with any single point $y \in X$, and then $\{\langle x_1, y\rangle, \langle x_2, y\rangle, \ldots \langle x_d, y\rangle\}$ is shattered by $C \times X$. Let $S = \{x_1, x_2, \ldots, x_d\}$. Since $S \times \{y\}$ is shattered by $C \times X$, for every $T \subseteq S$, there is a $c \in C$ such that $(c \times X) \cap (S \times \{y\}) = T \times \{y\}$. This is true if and only if $c \cap S = T$. Thus for every $T \subseteq S$, there is some $c \in C$ such that $c \cap S = T$ and thus S is shattered by C. Since $|S| = |S \times \{y\}|$, this demonstrates that $\text{VCdim}(C \times X) \leq \text{VCdim}(C)$, and thus completes the proof of the claim that $\text{VCdim}(C \times X) = \text{VCdim}(C)$.

Finally, for any concept classes C_1, C_2, define the *internal intersection* (denoted $\sqcap$ of C_1 and C_2 by $C_1 \sqcap C_2 = \{c_1 \cap c_2 : c_1 \in C_1, c_2 \in C_2\}$. Lemma 3.2.3 of (Blumer et al., 1987) shows that if C has VC dimension d, then $C \sqcap C$ has VC dimension at most $4d \log 6$. A virtually identical proof shows that $C_1 \sqcap C_2$ has VC dimension at most $4d \log 6$, for any two concept classes C_1, C_2 each with VC dimension d. This result, together with our claim above, shows that $(C \times X) \sqcap (X \times C)$ has VC dimension at most $4d \log 6$. To complete the proof of the lemma, note that $C \times C = (C \times X) \sqcap (X \times C)$; thus $\text{VCdim}(C \times C) \leq 4d \log 6$. □

As an example, the concept class of axis-aligned rectangles in the Euclidean plane satisfies the hypothesis of Theorem 6.8, and thus is ss-learnable.

As a final sufficient condition, we show that if C is polynomially learnable *from positive examples alone,* then C is ss-learnable. For simplicity of exposition, the definition below of learnability from positive examples is slightly nonstandard, although our results hold also for more standard definitions (for example, the unparameterized version of the definitions of Natarajan (1987), or Pitt and Valiant (1988)). It is essentially the same as the definition of polynomial learnability, but restricts access of the learning algorithm to positive examples only, and further requires that the concept it finds have perfect accuracy on the set of negative examples.

Definition 6.10. The concept class $C \subseteq 2^X$ is polynomially learnable *from positive examples alone* iff there exists an algorithm A and polynomials p and q such that for all $c \in C$, for every probability distribution D on elements of c (positive examples) and for all $\epsilon, \delta > 0$, if A is given as input $p\left(\frac{1}{\epsilon}, \frac{1}{\delta}\right)$ labeled examples generated by $\text{EXAMPLES}_{c,D}$, and the parameters ϵ and δ, then in time $q\left(\frac{1}{\epsilon}, \frac{1}{\delta}\right)$ A outputs the description of a concept $c' \in C$ such that with probability at least $1 - \delta$, $D(c - c') \leq \epsilon$ and $c' - c = \emptyset$.

Theorem 6.11. If C is polynomially learnable from positive examples alone, then C is ss-learnable.

Proof. By Theorem 6.8 (and the fact that learnability from positive examples alone trivially implies polynomial learnability), it suffices to show that if C is polynomially learnable from positive examples alone, then CC has a randomized polynomial time hypothesis finder. We describe a randomized polynomial time algorithm that, given as input a collection $S \subseteq X \times X$ of labeled examples of some concept $c \times c \in CC$, will output a concept $c' \times c' \in CC$ that is consistent with S.

Let S^+ consist of the positive examples of $c \times c$ in S, and let $m = |S^+|$. Note that if $\langle x, y\rangle \in S^+$ then $x \in c$ and $y \in c$ (whereas if $\langle x, y\rangle$ is a negative example of $c \times c$, we cannot deduce whether $x \notin c$, or $y \notin c$, or both). Form the set $P = \{x : \exists y$ such that $\langle x, y\rangle \in S^+$ or $\langle y, x\rangle \in S^+\}$. Let A be a learning algorithm for C that uses positive examples only. Now run A with accuracy parameter $\epsilon < \frac{1}{2m} \leq \frac{1}{|P|}$, and confidence parameter $\delta < \frac{1}{2}$. If a positive example is requested, randomly choose an element of P according to the (uniform) distribution D assigning each element of P probability $\frac{1}{|P|}$. By the definition of polynomial learning from positive examples alone, A will find, with probability at least $\frac{1}{2}$, a concept $c' \in C$ such that $D(c - c') \leq \epsilon < \frac{1}{2m}$ and $c' - c = \emptyset$. The first condition in fact asserts that c' contains each element of P, otherwise the error according to D would be at least $\frac{1}{|P|} \geq \frac{1}{2m}$, a contradiction. The second condition asserts that c' contains no element of $\bar{c}$. It follows that $c' \times c'$ is consistent with S. □

Finally, note that by Lemma 6.7, Theorems 6.8 and 6.11 show ss-learnability in a stronger sense; the time needed to ss-learn r disjoint concepts is polynomial in r as well as n, $\frac{1}{\epsilon}$, and $\frac{1}{\delta}$.

7. Equivalence of two types of learning

An interesting aspect of the definition of ss-learnability is that it is not at all clear how an algorithm might test a candidate solution for correctness. In concept learning, it is possible to test the accuracy of the learned concept using labeled examples of the unknown concept which are provided by the teacher. In ss-learnability, all that is available is a randomly generated pair, possibly totally unrelated to any examples seen before.

From this perspective, a reasonable alternate definition of ss-learning would only require that the algorithm find a set of formulas from the set F_n that correctly predicts (within ϵ) the labels from randomly generated LABELED-PAIRS, instead of requiring ϵ-closeness to some unkown formulas. The alternate definition is given below; "sc" stands for "same concept."

Definition 7.1. A family $F = \cup_{n\in N} F_n$ of Boolean formulas is *sc-learnable* iff for each number $r \in N$ there exists an algorithm A and polynomials p and q such that for all $n \geq 1$,, for every disjoint collection $\{f_1, f_2 \ldots, f_r\} \subseteq F_n$, for any probability distribution D on $\{0, 1\}^n$ satisfying equation (1), and for all $\epsilon, \delta > 0$, if A is given

as input the parameters ϵ and δ, and at most $p\left(n, \frac{1}{\epsilon}, \frac{1}{\delta}\right)$ labeled pairs generated from LABELED-PAIRS$_{f_1,f_2\ldots,f_r,D}$, then in time $q\left(n, \frac{1}{\epsilon}, \frac{1}{\delta}\right)$, A outputs a collection $\mathcal{G} = \{g_1, g_2 \ldots, g_s\}$ of (not necessarily disjoint) formulas in F_n that with probability at least $1 - \delta$ has the following property: The probability that a pair of examples drawn from LABELED-PAIRS$_{f_1,f_2\ldots,f_r,D}$ is incorrectly classified by $\mathcal{G}$ as to whether or not they are from the same concept is at most ϵ. (A pair is correctly classified by $\mathcal{G}$ if either the pair is $(x, y, \textit{same})$ and both x and y are in exactly one $g \in \mathcal{G}$, or the pair is $(x, y, \textit{different})$ and for some $g, g' \in \mathcal{G}$, $g \neq g'$, $x \in g$, $y \in g'$, and neither x nor y are in any other element of $\mathcal{G}$.)

Note that in the above definition, if a pair generated by LABELED-PAIRS contains some element x that is not contained in any element of $\mathcal{G}$, then this is counted as an incorrect classification. Similarly, if a pair contains an element in the intersection of two elements of $\mathcal{G}$, then this is also an incorrect classification.

Theorem 7.2. A family F is sc-learnable iff it is ss-learnable.

Proof. Suppose that F is ss-learnable. Then for any n, and any disjoint $f_1, f_2 \ldots, f_r \in F_n$, we can obtain with probability at least $1 - \delta$, and in time polynomial in n, $\frac{2}{\epsilon}$, and $\frac{1}{\delta}$, a set of (not necessarily disjoint) concepts $\mathcal{G} = \{g_1, g_2 \ldots, g_s\}$ such that $\mathcal{G}$ is $\frac{\epsilon}{2}$-close to $\{f_1, f_2 \ldots, f_r\}$. We show that using the learned concepts $\mathcal{G}$ to predict sameness/differentness will satisfy the requirements of sc-learnability.

Suppose that $\mathcal{G}$ is $\frac{\epsilon}{2}$-close to $\{f_1, f_2 \ldots, f_r\}$, that LABELED-PAIRS outputs the pair $(x, y, \textit{label})$, with $\textit{label} \in \{\textit{same}, \textit{different}\}$, and that $x \in f_x$ and $y \in f_y$, where $f_x, f_y \in \{f_1, f_2 \ldots, f_r\}$. Since $\mathcal{G}$ is $\frac{\epsilon}{2}$-close to $\{f_1, f_2 \ldots, f_r\}$, by definition there is a subset $\mathcal{G}' \subseteq \mathcal{G}$ and an injection $I : \mathcal{G}' \to \{f_1, f_2, \ldots, f_r\}$ such that with probability at least $1 - \frac{\epsilon}{2}$, x is in exactly one concept $g_x \in \mathcal{G}$, g_x is in $\mathcal{G}'$, and $x \in I(g_x)$ (and thus $I(g_x) = f_x$). The analogous relationships are true for y.

Case 1: $f_x = f_y$. With probability at least $1 - \epsilon$, x is in exactly one concept $g_x \in \mathcal{G}$, y is in exactly one concept $g_y \in \mathcal{G}$, g_x and g_y are in $\mathcal{G}'$, and $I(g_x) = f_x = f_y = I(g_y)$, so the learned concepts $\mathcal{G}$ produce the correct response of "same concept."

Case 2: $f_x \neq f_y$. With probability at least $1 - \epsilon$, x is in exactly one concept $g_x \in \mathcal{G}$, y is in exactly one concept $g_y \in \mathcal{G}$, g_x and g_y are in $\mathcal{G}'$, and $I(g_x) = f_x \neq f_y = I(g_y)$, so $\mathcal{G}$ produces the correct response of "different concepts."

Thus with probability at least $1 - \delta$, $\mathcal{G} = \{g_1, g_2 \ldots, g_s\}$ is $\frac{\epsilon}{2}$-close to $\{f_1, f_2 \ldots, f_r\}$ and the probability of correct classification is at least $1 - \epsilon$. Hence F is ss-learnable implies that F is sc-learnable.

Now suppose that F is sc-learnable. If F is pac-learnable as well, then we are done by using the sc-learned formulas as an oracle SC, and applying Theorem 4.2. However, it is not clear whether F is sc-learnable implies F is pac-learnable. (The obvious approach to showing this by letting $r = 1$ fails because there are then no negative examples, so nothing constrains the ss-learning algorithm from over-generalizing. If we let $r = 2$, with the second concept being the negative examples of some target concept to be pac-learned, then the *ss*-learning algorithm is only

required to work provided that the negative examples can also be expressed as a concept in F.) We show that regardless of the pac-learnability of F, if F is sc-learnable then F is ss-learnable.

Let F be sc-learnable. Then for any n, and any collection $\mathcal{F} = \{f_1, f_2 \ldots, f_r\}$ of disjoint elements of F_n, we can obtain in polynomial time, with probability at least $1 - \delta$, a collection $\mathcal{G} = \{g_1, g_2 \ldots, g_s\}$ of (not necessarily disjoint) elements of F_n such that the elements of $\mathcal{G}$ correctly classify pairs from LABELED-PAIRS as to sameness/differentness with probability at least $1 - \frac{\epsilon^2}{65r^2}$. We will show that $\mathcal{G}$ is therefore ϵ-close to $\mathcal{F}$, and the theorem follows.

Claim. For each $\frac{\epsilon}{4r}$-significant $f \in \mathcal{F}$ there is a unique $g \in \mathcal{G}$ such that $D(f \cap g) \geq \frac{\epsilon}{8r}$.

Proof of Claim. Assume there is *no* such g. Then the probability of choosing two points from f not both in some particular $g \in \mathcal{G}$ (and thus obtaining an incorrect classification) is at least

$$\frac{\epsilon}{4r}\left(\frac{\epsilon}{4r} - \frac{\epsilon}{8r}\right) = \frac{\epsilon^2}{32r^2} > \frac{\epsilon^2}{65r^2},$$

a contradiction. Now assume that there is more than one such element of $\mathcal{G}$, say g and g'. Then the probability of choosing two points x, y such that $x \in f \cap g$ and $y \in f \cap g'$ (and thus obtaining an incorrect classification regardless of whether g and g' are disjoint) is at least $\left(\frac{\epsilon}{8r}\right)^2 > \frac{\epsilon^2}{65r^2}$, a contradiction, thus proving the claim.

□

To complete the proof of Theorem 7.2, we find a subset $\mathcal{G}' \subseteq \mathcal{G}$ and a bijection $I : \mathcal{G}' \to \{f \in \mathcal{F} : f \text{ is } \frac{\epsilon}{4r}\text{-significant}\}$ (and hence an injection with range $\mathcal{F}$) witnessing that $\mathcal{G}$ is ϵ-close to $\mathcal{F}$. For each $\frac{\epsilon}{4r}$-significant f, let the unique g such that $D(f \cap g) \geq \frac{\epsilon}{8r}$ be an element of $\mathcal{G}'$, with $I(g) = f$.

Now for each $g \in \mathcal{G}'$, $D(g \oplus I(g)) \leq \frac{\epsilon}{4r}$; for if not, then either $D(g - I(g)) \geq \frac{\epsilon}{8r}$ or $D(I(g) - g) \geq \frac{\epsilon}{8r}$. The cases are similar; we show that the first case cannot happen: If

$$D(g - I(g)) = D(g \cap \overline{I(g)}) \geq \frac{\epsilon}{8r},$$

then since $D(g \cap I(g)) \geq \frac{\epsilon}{8r}$ (by definition of g), we have the probability that a misclassification occurs is at least $\left(\frac{\epsilon}{8r}\right)^2 > \frac{\epsilon^2}{65r^2}$, a contradiction.

It follows that

$$D(E2) = D(\bigcup_{g_i \in \mathcal{G}'} g_i \oplus I(g_i)) \leq r\frac{\epsilon}{4r} = \frac{\epsilon}{4}.$$

Then, as in the proof of Lemma 3.7,

$$E1 - E2 \subseteq \bigcup \{f_i : f_i \text{ is } \epsilon/4r\text{-insignificant}\}, \qquad [6]$$

which is true if

$$E1 \cap \bigcup \{f_i : f_i \text{ is } \epsilon/4r\text{-significant}\} \subseteq E2. \qquad [7]$$

To see that containment (7) holds, note that if $x \in f_i$ and f_i is $\frac{\epsilon}{4r}$-significant, then by the claim, there is a unique $g_j \in \mathcal{G}'$ such that $D(f_i \cap g_j) \geq \frac{\epsilon}{8r}$, and thus $I(g_j) = f_i$. If x is also in $E1 = \overline{\cup \mathcal{G}'}$, then $x \notin g_j$, so $x \in g_j \oplus I(g_j) \subseteq E2$, and containment (7) follows. Containment (6) implies that

$$D(E1 - E2) \leq \frac{\epsilon}{4}.$$

By the definition of sc-learnability,

$$D(E3) = D(\bigcup_{g_i, g_j \in \mathcal{G}, i \neq j} (g_i \cap g_j)) \leq \frac{\epsilon^2}{65r^2} < \frac{\epsilon}{2},$$

so

$$D(E1 \cup E2 \cup E3) \leq D(E2) + D(E1 - E2) + D(E3) < \frac{\epsilon}{4} + \frac{\epsilon}{4} + \frac{\epsilon}{2} = \epsilon.$$

□

Corollary 7.3. Each of the families described in Theorems 3.1 and Corollaries 5.1, 5.2, and 5.3 is sc-learnable. Further, any concept class satisfying the hypothesis of Theorems 4.2, 6.8, or 6.11, or Corollary 6.3 is sc-learnable.

8. Conclusion

We have defined a new type of learning in which multiple concepts are learned simultaneously. This is done, furthermore, using a teacher that provides less information than the teacher in pac-learning. This definition provides for a level of

teacher involvement that is intermediate between supervised and unsupervised learning.

We have shown that a number of pac-learnable concept classes are also learnable under the new definition; we leave open the question of whether this is true of all pac-learnable classes. It is also not known whether the set of ss-learnable classes is a subset of the pac-learnable classes; that is, whether there exist concept classes that are not pac-learnable but are ss-learnable. As noted in the proof of Theorem 7.2, trying to pac-learn a single concept by ss-learning the concept with $r = 1$ fails to resolve this issue.

A main open problem is to determine whether there is an ss-learning algorithm for monomials (or any of the other classes considered here) with running time polynomial in the parameter r, the number of disjoint classes.

Another potential area of study is the effect of removing the restriction on the distribution D that requires all of the points generated by LABELED-PAIRS to come from one of the concepts to be learned. The definition of LABELED-PAIRS would have to be altered to reflect the fact that points in the pairs may not be in any of the concepts. Allowing arbitrary distributions would, in addition to making the definition more general, cause the case of ss-learning a single concept to more closely approximate single concept learning, since negative examples of the concept would be permitted.

Haussler (1988) shows that internal disjunctive concepts are pac-learnable. It can be shown that they are also ss-learnable, provided that the length of the compound terms can be bounded by a constant. An interesting open problem is whether the class is still ss-learnable when this restriction is lifted.

Our technique of constructing an intermediate oracle to facilitate learning may prove useful in other learning problems. The use of such an oracle enables us to exploit the phenomenon of concept classes that are learnable by other classes, but are not themselves learnable.

Acknowledgements

This research was supported in part by NSF Grant IRI-8809570. The authors are grateful to Robert Reinke for posing the problem addressed in this paper. This paper revises and corrects a preliminary version, which appeared as Technical Report UIUCDCS-R-87-1372, Department of Computer Science, University of Illinois at Urbana-Champaign.

Notes

1. In some applications it may be desirable to parameterize a family F by some size measure other than (or in addition to) the number of variables.
2. See Haussler, Kearns, Littlestone, & Warmuth (1988), Littlestone (1988), and Pitt & Warmuth (1988) for comparisons of this and other models of prediction.

References

Anderberg, M. (1973). *Cluster analysis for applications.* New York: Academic Press.

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

Angluin, D. (1988a). Queries and concept learning. *Machine Learning, 2,* 319–342.

Angluin, D. (1988b). Learning with hints. *Proceedings of the 1988 Workshop on Computational Learning Theory* (pp. 167–181). Cambridge, MA: Morgan Kaufmann.

Angluin, D. (1988c). *Negative results for equivalence queries.* (Technical Report YALEU/DCS/RR-648). New Haven, Connecticut: Yale University, Department of Computer Science.

Angluin, D. (1988d). *Equivalence queries and DNF formulas.* (Technical Report YALEU/DCS/RR-659). New Haven, Connecticut: Yale University, Department of Computer Science.

Berman, P., & Roos, R. (1987). Learning one-counter languages in polynomial time. *Proceedings of the 28th IEEE Symposium on Foundations of Computer Science* (pp. 61–67). Los Angeles: IEEE Computer Society Press.

Blumer, A., Ehrenfeucht, A., Haussler, D., & Warmuth, M. (1986). Classifying learnable geometric concepts with the Vapnik-Chervonenkis dimension. *Proceedings of the 18th Annual ACM Symposium on Theory of Computation* (pp. 273–282). Berkeley, CA: Association for Computing Machinery.

Blumer, A., Ehrenfeucht, A., Haussler, D., & Warmuth, M. (1987). *Learnability and the Vapnik-Chervonenkis dimension.* (Technical Report UCSC-CRL-87-20). Santa Cruz, CA: University of California, Santa Cruz.

Carbonell, J. G., Michalski, R. S. & Mitchell, T. M. (1983). An overview of machine learning. In R. S. Michalski, J. G. Carbonell & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach.* Palo Alto, CA: Tioga Press.

Duda, R., & Hart, P. (1973). *Pattern classification and scene analysis.* New York: John Wiley & Sons.

Garey, M., & Johnson, D. (1979). *Computers and intractability: A guide to the theory of NP-completeness.* San Francisco: W. H. Freeman.

Hartigan, J. (1975). *Cluster algorithms.* New York: John Wiley & Sons.

Haussler, D. (1988). Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial Intelligence, 36,* 177–221.

Haussler, D., Kearns, M., Littlestone, N., & Warmuth, M. (1988). Equivalence of models for polynomial learnability. *Proceedings of the 1988 Workshop on Computational Learning Theory* (pp. 42–55). Cambridge, MA: Morgan Kaufmann.

Haussler, D., Littlestone, N., & Warmuth, M. (1988). Predicting {0, 1} functions on randomly drawn points. *Proceedings of the 29th Annual Symposium on Foundations of Computer Science* (pp. 100–109). White Plains, NY: IEEE Computer Society Press.

Kearns, M., Li, M., Pitt, L., & Valiant, L. G. (1987a). On the learnability of Boolean formulae. *Proceedings of the 19th Annual ACM Symposium on Theory of Computing* (pp. 285–295). New York: Assoc. Comp. Mach.

Kearns, M., Li, M., Pitt, L., & Valiant, L. G. (1987b). Recent results on Boolean concept learning. *Proceedings of the Fourth International Workshop on Machine Learning.* Irvine, CA: Morgan Kaufmann.

Littlestone, N. (1988). Learning quickly when irrelevant attributes abound: a new linear threshold algorithm. *Machine Learning, 2,* 285–319.

Natarajan, B. K. (1987). On learning Boolean functions. *Proceedings of the 19th Annual ACM Symposium on Theory of Computing* (pp. 296–304). New York: Assoc. Comput. Mach.

Pitt, L., & Valiant, L. G. (1988). Computational limitations on learning from examples. *Journal of the ACM, 35,* 965–984.

Pitt, L., & Warmuth, M. K. (1988). Reductions among prediction problems: on the difficulty of predicting automata. *Proceedings of the Third Annual Conference on Structure in Complexity Theory* (pp. 60–69). Washington, D.C.: IEEE Computer Society Press.

Rivest, R. (1987). Learning decision-lists. *Machine Learning, 2,* 229–246.
Romesburg, H. (1984). *Cluster analysis for researchers.* Belmont, CA: Lifetime Learning.
Valiant, L. G. (1984). A theory of the learnable. *CACM, 27,* 1134–1142.
Valiant, L. G. (1985). Learning disjunctions of conjunctions. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*(pp. 560–566). Los Angeles, CA: Morgan Kaufmann.