Results on learnability and the Vapnik-Chervonenkis dimension*

(Extended Abstract)

Nathan Linial[†] Yishay Mansour[‡] Ronald L. Rivest[§]

Abstract

We consider the problem of learning a concept from examples in the distribution-free model introduced by Valiant[13]. (An essentially equivalent model, if one ignores issues of computational difficulty, was studied by Vapnik and Chervonenkis [14].) We introduce the notion of *dynamic sampling*, wherein the number of examples examined may increase with the complexity of the target concept. This method is used to establish the learnability of various concept classes with an *infinite* Vapnik-Chervonenkis dimension. We also discuss an important variation on the problem of learning from examples, called *approximating from examples*. Here we do *not* assume that the target concept T is a member of the concept class C from which approximations are chosen. This problem takes on particular interest when the VC dimension of C is infinite. Finally, we discuss the problem of computing the VC dimension of a finite concept set defined on a finite domain.

1 Introduction

1.1 Learning in the Distribution-Free Model

In this model, each concept C is a subset of a given instance space X. For example, X might be $\{0,1\}^n$ or real n-dimensional space R^n. (In some cases it is more natural to use $\cup_n\{0,1\}^n$ or $\cup_n R^n$ so that each instance is an n-bit vector or a vector of n reals where n is *arbitrary*; we don't consider such variations here.) The class of concepts being learned will be denoted $\mathcal{C}$. The unknown target concept T to be learned is assumed to be a member of $\mathcal{C}$.

In this model there is a fixed but arbitrary probability function P defined on X. The learning algorithm L will have access to a source of examples of the unknown target concept T. Each time the algorithm obtains an example from this source it draws an element $x \in X$ independently according to P. This is sometimes called the "one-oracle" or the "one-button" model. (In a "two-button" model the learning algorithm can request either a positive or a negative example, and these examples are produced according to separate probability distributions.) In this paper we assume that every concept C (including the target concept) has a well-defined probability with respect to P; we do not address issues of measurability, etc., here.

Each example (x, c) consists of an instance x and its classification $c \in \{+,-\}$ as either a positive instance ($x \in T$) or a negative instance ($x \notin T$) of the unknown target concept T.

In addition to the source of the examples, the

[*] This paper was prepared with support from NSF grant DCR-8607494, ARO Grant DAAL03-86-K-0171, and the Siemens Corporation.

[†] IBM Research Center - Almaden, 650 Harry Road, San Jose, CA 95120 and Computer Science Dept., Hebrew University, Jerusalem Israel; e-mail: nati@ibm.com

[‡] Laboratory for Computer Science, Massachusetts Institute of Technology Cambridge, MA 02139; e-mail: mansour@theory.mit.edu

[§] Laboratory for Computer Science, Massachusetts Institute of Technology Cambridge, MA 02139; e-mail: rivest@theory.mit.edu

0272-5428/88/0000/0120$01.00 © 1988 IEEE

learning algorithm takes as input two parameters: ϵ (the *accuracy* parameter) and δ (the *confidence* parameter). After drawing a number of samples, the learning algorithm produces as output a description of a concept C, which may be different than the true target concept T. (Usually it is required that $C \in \mathcal{C}$, although other restrictions on C are sometimes considered.) The error rate of a concept C (with respect to the true concept T and the probability distribution P) is $P(C \oplus T)$, the probability that C and T classify a randomly drawn example differently. (Here $C \oplus T = (C - T) \cup (T - C)$, the symmetric difference of C and T.) We say that the concept C output by the learning algorithm is *approximately correct* if the error rate $P(C \oplus T)$ is at most ϵ, If (for any concept class $\mathcal{C}$, probability distribution P, accuracy parameter ϵ and confidence parameter δ), the probability that the output is approximately correct is at least $1 - \delta$, we say that learning algorithm is *probably approximately correct*; such a learning algorithm is said to *pac-learn* the concept class $\mathcal{C}$, and $\mathcal{C}$ is said to be *pac-learnable.*

The learning algorithm L requires two sorts of resources: computational time and examples; we define the *time complexity* and the *sample complexity* of L to be amount of each resource used.

We say that L is a *polynomial pac-learning algorithm*, and that the class $\mathcal{C}$ is *polynomially pac-learnable*, if L pac-learns $\mathcal{C}$ with time complexity and sample complexity which are polynomial in $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$.

Normally, a polynomial pac-learning algorithm can be made to run in *static sampling mode*, where a sample containing all of the necessary examples is drawn before any computation is performed. Typically, pac-learning algorithms are *consistent* in that the concept C they return agrees with the classification of each example of the sample.

There are models of learning other than oracle-based models. For example, a "functional model" has recently been shown to be equivalent to the oracle model by Haussler et. al. [6]; their paper contains a wealth of information about different models of learning.

In this paper we will be concerned almost exclusively with the sample complexity of learning algorithms.

1.2 Previous Results about Distribution-Free Learning

In order for a pac-learning algorithm to pac-learn a concept class $\mathcal{C}$, the measured error rate of each concept $C \in \mathcal{C}$ (measured on the examples seen) must be a good estimate of the true error rate $P(T \oplus C)$ of that concept. Vapnik and Chervonenkis [14] were able to find a nice characterization of classes of concepts $\mathcal{C}$ for which the measured error rate converges *uniformly* (over $\mathcal{C}$) to the true error rate.

For this purpose they introduce the notion of a dimension (usually called the Vapnik-Chervonenkis dimension, or VC dimension) of a concept class and showed that a sufficient condition for the uniform convergence of the measured error rates to the true error rates is that the Vapnik-Chervonenkis dimension of $\mathcal{C}$ be finite.

Definition 1.1 *The Vapnik-Chervonenkis dimension of a concept class $\mathcal{C}$ is the largest d such that there exists a set S of d instances, such that for every subset $U \subseteq S$ there exists a concept $C \in \mathcal{C}$ such that $U = S \cap C$.*

Their work has been extended to handle much more general situations; see Pollard [9] for a nice exposition.

Blumer et. al. [3] were the first to draw the connection between distribution free learning and the VC dimension. They gave bounds on the number $m(\epsilon, \delta)$ of examples needed by a consistent pac-learning algorithm to pac-learn a concept class $\mathcal{C}$, in terms of $\mathcal{C}$'s VC dimension d:

$$\begin{aligned} m(\epsilon,\delta) &= O(\frac{1}{\epsilon}\ln\frac{1}{\delta} + \frac{d}{\epsilon}\log\frac{1}{\epsilon}), \\ m(\epsilon,\delta) &= \Omega(\frac{1}{\epsilon}\ln\frac{1}{\delta} + d)\ . \end{aligned}$$

Specifically, the upper bound proved is:

$$m(\epsilon,\delta) \le \max(\frac{4}{\epsilon}\ln\frac{2}{\delta}, \frac{8d}{\epsilon}\ln\frac{13}{\epsilon}), \qquad (1)$$

we shall use this result later. These results imply the following.

Theorem 1.1 [Blumer, Ehrenfeucht, Haussler, and Warmuth [3]] *A concept class C is pac-learnable with static sampling if and only if C has finite VC dimension.*

The lower bound was improved by Ehrenfeucht et. al. [5] to

$$m(\epsilon, \delta) = \Omega(\frac{1}{\epsilon} \ln \frac{1}{\delta} + \frac{d}{\epsilon}) . \qquad (2)$$

Based on their upper bound, Blumer et. al. also showed many concept classes to be polynomially pac-learnable. More generally, if C is a class with finite VC dimension and there exists a polynomial time algorithm to find a concept in C that is consistent with a given sample, then C is polynomially pac-learnable. Polynomial pac-learning algorithms have been developed for specific problems such as k-CNF [13] and decision lists [12]. See [7,8] for surveys of other known results in this area.

1.3 Our contributions

We introduce the notion of *dynamic sampling*, wherein the number of examples examined increases with the complexity of the target concept. It turns out that dynamic sampling does indeed enrich the class of pac-learnable concepts, compared to static sampling. We show examples of concept classes which our scheme pac-learns (using dynamic sampling) despite the fact that the class has an infinite VC dimension and hence can not be pac-learned with static sampling.

Similar results and techniques have recently (and independently) been obtained by others. For example, the notion of dynamic sampling appears in a proof by Haussler et. al. [6] that the size of target concept need not be known if one is willing to sacrifice the requirement that the learning algorithm should always halt. Based on this result, Blumer et. al. [4] and have shown how to pac-learn concept classes of infinite VC dimension. (There are some minor technical differences; while the overall approaches are essentially identical, we present effective techniques for minimizing the number of stages required by dynamic sampling; an issue they do not address. Also, our pac-learning algorithms always terminate whereas theirs halt only probabilistically.) Benedek and Itai [2] give similar results in a slightly different model.

Next we discuss another important variation on the problem of learning from examples. This is the problem of *approximating from examples.* Here we do *not* assume that the target concept T is a member of the concept class C from which approximations are chosen. This problem takes on particular interest when the VC dimension of C is infinite.

Finally, we state the problem of computing the VC dimension for a finite domain as a combinatorial problem, called the "discrete VC problem". The discrete VC problem can be easily solved in time $O(rn^{\lg r})$, where n is the number of points and r the number of concepts.[1] It is not known if this problem is in P; it may be NP-complete. We give a combinatorial characterization for classes with VC dimension 1. It is an open problem to characterize classes of higher VC dimensions.

2 The idea of dynamic sampling

We begin with a proof that any enumerable concept class is pac-learnable using dynamic sampling; this example illustrates the power of dynamic sampling relative to static sampling, since an enumerable class of concepts may have infinite VC dimension.

When using dynamic sampling, the pac-learning algorithm alternates between drawing examples and doing computations. A *stage* of the pac-learning algorithm consists of drawing a set of examples and performing the subsequent computations. Pac-learning a class with infinite VC dimension may require an unbounded number of stages, as we shall show in section 2.3.

2.1 An example – learning an enumerable concept class

Let $C = \{C_1, C_2, \ldots\}$ be an (recursively) enumerable concept class, such that for each C_i mem-

[1] We use $\lg x$ to denote $\log_2 x$ throughout.

bership in C_i is decidable. Note that C may have infinite VC dimension. (For example, let N be the set of natural numbers, and let C be the set of all finite subsets of N.) We show that C is pac-learnable as follows.

Algorithm "Enumerable-Learner":

```
1. Let $i = 1$.
2. Draw enough examples so that the total number $m_i$ of examples drawn so far is at least $\frac{1}{\epsilon}\ln\frac{2i^2}{\delta}$.
3. If $C_i$ is consistent with all examples seen so far then output $C_i$. Otherwise increase $i$ by 1 and return to step 2.
```

Theorem 2.1 Enumerable-Learner *can pac-learn any enumerable concept class C.*

Proof: Let T be the target concept. Call concept C_i "ϵ-bad" if $P(T \oplus C_i) \geq \epsilon$. The probability that an ϵ-bad concept C_i is output is at most $(1-\epsilon)^{m_i}$, since this is an upper bound on the probability that C_i is consistent with the m_i examples seen so far. For $m_i > \frac{1}{\epsilon}\ln\frac{2i^2}{\delta}$ the inequality $(1-\epsilon)^{m_i} < \frac{\delta}{i^2}\frac{6}{\pi^2}$ holds. Since

$$\sum_{i=1}^{\infty}\frac{1}{i^2} = \frac{\pi^2}{6}, \tag{3}$$

the probability that Enumerable-Learner outputs an ϵ-bad concept is at most

$$\frac{6}{\pi^2}\sum_{i=1}^{\infty}\frac{\delta}{i^2} = \delta . \tag{4}$$

□

A similar scheme appears in [7] with $m_i > \frac{i}{\epsilon} + \frac{1}{\epsilon}\ln\frac{1}{\delta}$. Note that our bound on m_i is *linear* in the logarithm of the index of the target concept.

2.2 Decomposable concept classes

The result proved above does not handle concept classes which are uncountable. In this subsection we extend our result by introducing the notion of a *decomposable* concept class. (This notion was proposed independently by Benedek and Itai[2].)

Definition 2.1 *A concept class C is* decomposable *if it can be written as the countable union*

$$C = C_1 \cup C_2 \cup \ldots$$

where each concept subclass C_d has VC dimension at most d.

(It is equivalent to require merely that each C_i have finite VC dimension.)

In many cases, this decomposition can be done in such a way that $C_i \subseteq C_{i+1}$ for all i, and those concepts in $C_d - C_{d-1}$ can naturally be said to have *size d*. For example, if each concept is represented by a binary string, we might let C_d be the set of concepts whose binary representation contains at most $d-1$ bits. Or, if $X = [0,1]$ and C is all finite unions of subintervals of X, then C_d can be the set of concepts which are the union of at most $d/2$ subintervals of $[0,1]$. In other cases, the "natural" size measure might be polynomially related to d; our results can be easily extended to handle these cases.

We would like the complexity of a pac-learning algorithm to be polynomial in the size of the target concept T being learned, where

$$size(T) = \min\{d : T \in C_d\} .$$

One way to accomplish this is to provide $size(T)$ as an additional input to the pac-learning algorithm. This reduces the problem to one of pac-learning the concept class $C_1 \cup C_2 \ldots C_{size(T)}$, which has finite VC dimension. However, this may be impossible in practice, so we rule this possibility out. We wish to have the pac-learning algorithm *determine* $size(T)$ itself.

Definition 2.2 *A concept class C is* uniformly decomposable *if it is decomposable and there exists an algorithm A which given an index d and a sample, can produce a concept $C \in C_d$ consistent with the sample, or else outputs "none" if no such concept exists. Furthermore, if the algorithm A runs in time polynomial in d and the number of examples in the sample we say that C is* polynomially uniformly decomposable.

The following algorithm can pac-learn any uniformly decomposable concept class:

Algorithm "Uniformly-Decomposable-Learner":

1. Let $d = 1$.

2. Draw enough samples so that the total number m_d of examples drawn so far is at least
$$\max(\frac{4}{\epsilon}\ln\frac{8d^2}{\delta}, \frac{8d}{\epsilon}\ln\frac{13}{\epsilon})$$

3. If there is a $C \in \mathcal{C}_d$ which is consistent with all examples seen so far then output C. Otherwise increment d by 1 and return to step 2.

Theorem 2.2 *Any uniformly decomposable concept class is pac-learnable, using dynamic sampling (procedure* **Uniformly-Decomposable-Learner***). If the class is polynomially uniformly decomposable, then the time and sample complexity are polynomial in the size of the target concept being learned.*

Proof: First we argue that the algorithm pac-learns any uniformly decomposable concept class. The reason is that *if* a concept in the class is found to be consistent with the data, then with high probability it is not ϵ-bad. Summing over all d, we show (as for the Enumerable-Learner), that the total probability of outputting a concept which is ϵ-bad is at most δ.

The number of examples at each stage, m_d, is chosen according to equation (1) with confidence parameter $\frac{\delta}{4d^2}$ and accuracy parameter ϵ. Therefore, the probability that an ϵ-bad concept is output in step 3, for a given value of d, is at most $\frac{\delta}{4d^2}$. Summing over all possible values of d, we get that the probability of an ϵ-bad concept to be output is bounded by δ. It remains to show that the algorithm terminates.

The value of d is incremented by 1 every stage, therefore after $size(T)$ stages, where T is the target concept, either the algorithm has terminated or $d = size(T)$. When $d = size(T)$, there is a consistent concept in $\mathcal{C}_d$ (i.e., T), so the algorithm terminates at this stage. Furthermore, the number of examples seen by the algorithm is polynomial in $size(T)$. For the case that the concept classes that polynomially uniformly decomposable, the running of the algorithm is polynomial in $size(T)$ as well. □

As an illustration of the power of these techniques, the following classes are pac-learnable, even though they are uncountable and have infinite VC dimension.

1. The concept class $\mathcal{C}_{FI}$ whose members are finite unions of subintervals of $[0, 1]$.

2. The concept class $\mathcal{C}_{PR}$ whose members are regions in the two-dimensional Euclidean plane defined by an inequality of the form $y \leq f(x)$, where f is any polynomial of finite degree with real coefficients.

3. The concept class $\mathcal{C}_{TC}$ whose members are defined by multilayer threshold circuits of arbitrary (finite) size and configuration in n-dimensional Euclidean space.

We show that the first problem is also polynomially pac-learnable. We are not sure about the complexity of the second problem, and we conjecture that the third problem is intractable.

Theorem 2.3 *The concept class $\mathcal{C}_{FI}$ is polynomially uniformly decomposable.*

Proof: Decompose the class $\mathcal{C}_{FI}$ such that each C_i includes all the concepts with at most $\frac{i}{2}$ subintervals. To show that $\mathcal{C}_{FI}$ is uniformly decomposable we need to exhibit an algorithm, A_{FI}, that given a sample and an index d, finds a consistent concept from $\mathcal{C}_d$ (if one exists) or output "none" (if such a concept does not exists). Algorithm A_{FI} is based on the observation that the number of alternations (from positive examples to negative examples or vice versa) in a sample along the interval $[0, 1]$ is at most twice the number of subintervals in the target concept. If the number of alternations is greater than d then A_{FI} outputs "none", else it outputs a concept with minimal number of subintervals that is consistent with the sample. (Clearly, the output concept is in $\mathcal{C}_d$.) Since A_{FI} runs in polynomial time in the sample size, $\mathcal{C}_{FI}$ is polynomially uniformly decomposable. □

Corollary 2.1 *The concept class C_{FI} is polynomially pac-learnable.* □

Our approach doesn't seem to help much for answering some open questions, such as whether DNF is polynomially pac-learnable. While the approach presented above (e.g. **Enumerable-Learner**) can be used to show that the sample complexity for pac-learning DNF is not too great, questions of computational complexity still remain unsolved.

The converse to Theorem 2.2 holds for classes that are polynomially pac-learnable. In such a case there is a polynomial (in $size(c)$) upper bound on the number of examples that we can draw while pac-learning a specific concept c. We can use the decomposition induced by the function *size* and refine it such that it will meet our definition of a decomposable class.

However, Benedek [1] has shown that the converse to Theorem 2.2 does not hold in general. That is, there are concept classes that are not decomposable but are pac-learnable (using dynamic sampling). His proof is based on the class C_{CI} whose members are countable unions of subintervals of $[0,1]$. In [2] it was shown that C_{CI} is not decomposable. The crux of his learning algorithm is that any concept $c_1 \in C_{CI}$, has a concept $c_2 \in C_{FI}$, such that $P(c_1 \oplus c_2) \leq \epsilon$.

It remains as an open problem to find a concept class that is not pac-learnable in our model.

Another open question is that of *consistency*: a learning algorithm is said to be consistent if its output always correctly classifies all of the examples the learning algorithm has seen. While consistency is not required of a learning algorithm, Haussler et. al. [6] have proven that there exists a general procedure for transforming an inconsistent static sampling procedure to a consistent static sampling procedure. Their proof does not apply to dynamic sampling algorithms, and it remains an open question as to whether such a transformation is possible.

2.3 The number of stages

A stage of a dynamic sampling pac-learning algorithm consists of drawing a sample and then doing a computation. The number of stages is the number of times a sample is drawn. In **Uniformly-Decomposable-Learner** the number of stages may be as large as $n = size(T)$. We next show that this can be improved for the concept class C_{FI}.

Theorem 2.4 *To pac-learn the concept class C_{FI} the number of stages required tois is $O(\lg\lg n)$.*

Proof: In each stage update the value of d to d^2 rather than $d+1$. (Note that the decomposition of C_{FI} is such that if $i < j$ then $C_i \subset C_j$.) □

For the class C_{FI} we now prove that this bound is tight.

Theorem 2.5 *Any algorithm that pac-learns C_{FI}, with respect to the uniform distribution using a number of examples that is bounded by a polynomial in the number n of subintervals of the target concept requires at least $\Omega(\log\log n)$ stages.*

Proof: First we show that $\Omega(n)$ examples are required for a worst case concept with at most n subintervals. (We can not use a VC dimension argument (e.g., equation (2)) since the probability distribution is fixed.) Divide the segment [0,1] into n equal length subintervals. With probability 1/2, every point in subinterval i will be a positive instance of the target concept, and with probability 1/2 every point in subinterval i will be a negative instance of the target concept, independently for each i. Clearly, the target concept can be expressed as a union of at most n subintervals. If the pac-learning algorithm draws less then $n/2$ examples, then the expected error rate is at least 1/4. However, if the total error rate exceeds $\delta + (1-\delta)\epsilon$ then pac-learning can not be achieved. By choosing, say, $\epsilon = \delta = 1/8$, we force the pac-learning algorithm to draw at least $n/4$ examples for some concept.

Since the algorithm uses a polynomial size sample, there are constants e and c such that the number of examples of the algorithm is bounded above by cn^e. Without loss of generality both e and c are greater than 1. (In general, c may depend on ϵ and δ. Since both ϵ and δ are fixed, we can regard c as a constant.). Let S_i be the number of examples drawn by the end of stage i. Note that there is a concept in C_{FI}

consistent with the samples drawn up to stage i which has at most S_i subintervals. Thus, if the algorithm proceeds to stage $i + 1$, it can draw at most $c{S_i}^e - S_i$ additional examples in stage $i + 1$, otherwise the bound on the sample size may be violated. Therefore, $S_{i+1} \leq c{S_i}^e$, so we can bound S_i from above by $c^{e^{2i}}$. Note that $S_{o(\log_e \log_c n)} = o(n)$. We assumed that the number of examples is at least $n/4$, hence, the number of stages is $\Omega(\log\log n)$. □

Finally, we show that not every concept class of infinite VC dimension requires an unbounded number of stages.

Theorem 2.6 *Let C_N denote the concept class of all subsets of the natural numbers. Then C_N can be pac-learned with a two-stage learning algorithm.*

Proof: In the first stage we draw a sample of size $(2/\epsilon)\lg(2/\delta)$. Let M denote the largest integer appearing in this sample. With probability at least $1 - \delta/2$ the probability associated with integers greater than M is at most $\epsilon/2$. In the second stage we consider the induced problem of learning the restriction of the target concept to the natural numbers at most M. This reduces the problem to one having a finite VC dimension (i.e., M), which can be solved with a static sampling algorithm with parameters $\epsilon/2$ and $\delta/2$. □

A simple generlization of this argument applies in a straightforward manner whenever the instance space is countable.

3 Approximating from examples

The problem of "learning from examples" is to find a good approximation for the target concept T, *given that T is from the class $\mathcal{C}$*. This assumes that we have *a priori*, or background, knowledge that the target concept is indeed from the class $\mathcal{C}$. This assumption may often be unrealistic. It is often more natural to assume that $\mathcal{C}$ contains concepts which may be "close" to T, even though T itself might not be a member of $\mathcal{C}$.

How should one proceed if it is not known *a priori* that $T \in C$? The algorithms given in the literature, and those in the previous sections, are built around the assumption that $T \in C$, so that they are guaranteed that there will always be a concept in $\mathcal{C}$ with zero true error rate (and thus zero error rate as measured on the examples seen). If $T \notin \mathcal{C}$, there may be *no* concept in $\mathcal{C}$ which has zero measured error rate. Even worse, it may be the case that every concept in $\mathcal{C}$ has true error rate greater than some fixed value $\beta > 0$. In what follows let

$$\beta = \inf_C\{P(C \oplus T) : C \in \mathcal{C}\} .$$

We therefore define the problem of "approximating T using $\mathcal{C}$ from examples" as the problem of producing, given as input

1. a source of examples labeled according to the target concept T,

2. an accuracy parameter ϵ, and

3. a confidence parameter δ,

a concept $C \in \mathcal{C}$ whose error rate is at most $\beta + \epsilon$, with probability at most $1 - \delta$. An algorithm for solving this problem is called an *approximation algorithm* for the class $\mathcal{C}$. Note that β is *not* an input to the approximation algorithm. If an approximation algorithm exists for $\mathcal{C}$ then we say that $\mathcal{C}$ is an *approximation class.* If this procedure runs in time polynomial in $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, and the size of C then we say that $\mathcal{C}$ is a *polynomial approximation class.*

When $T \in C$, we have $\beta = 0$, and this problem reduces to the problem of pac-learning from examples.

When $\mathcal{C}$ has finite VC dimension, then $\mathcal{C}$ is an approximation class. (This result follows simply from the results of Vapnik and Chervonenkis [14].) The procedure is merely to draw enough examples, and then return the concept with the lowest measured error rate.

What happens when $\mathcal{C}$ has infinite VC dimension?

Suppose further that $\mathcal{C}$ is "strongly uniformly decomposable" in that it is decomposable and there exists a procedure which, given as inputs d and a sample, returns the concept in $\mathcal{C}_d$ which has minimum error rate on the sample.

Can we then modify our procedure **Uniformly-Decomposable-Learner** to show that C is an approximation class in general?

The answer seems to be *no.* Let us define

$$\beta_d = \inf_{C_d}\{P(C \oplus T) : C \in C_d\} \ .$$

and consider the case that

$$\beta_d = \epsilon + \frac{1}{d} > \epsilon \ .$$

As we go to larger and larger d's, the true error rate for the best concept of size d gets smaller and smaller. The problem is, how can the learner predict that $\beta \neq 0$?

We face here the problem of trading off between "complexity of the hypothesis" (i.e., concept size) and "fit to the data" (i.e., true error rate). The problem of striking this tradeoff well is a classic one – a standard example is the problem of fitting a polynomial to a set of points representing noisy measurements, where the degree of the polynomial is unspecified. The tradeoff is often achieved by minimizing a function which is the sum of some function of the concept size and some function of the amount of information needed to describe all of the classifications in the sample, given a description of the concept for free. (This is the "minimum description length principle" as proposed by Rissanen [11]; see also Quinlan and Rivest[10].) However, in our case the size of the sample is not fixed, but is up to the learning algorithm; thus the MDLP approach is not applicable.

In some cases it may be possible to estimate β well from the data. For example, if C consists of all finite subsets of the integers which contain the integer 1, (assuming that $T \notin C$) then β is just $P(1)$, which can be estimated from the observed frequency of 1 in the data. This class has infinite VC dimension but is a polynomial approximation class. In general, however, the value of β is not easy to estimate, and the algorithm is faced with the problem of trying to guess whether by increasing d sufficiently the value of β_d will drop. Conversely, we note that if C is an approximation class, then β can be estimated accurately.

It may be that this problem may only be tractable in general when the algorithm is also given as input an upper bound $\hat{\beta} \geq \beta$, and the algorithm must produce as output a concept C whose true error rate is at most $\hat{\beta} + \epsilon$, with high probability. The techniques we have given previously can be modified to handle this case. (The problem of pac-learning from examples is essentially the case $\hat{\beta} = 0$.)

Thus we see that while pac-learning from examples generalizes nicely to the case of infinite VC dimension, approximating from examples doesn' seem to generalize as well. This is unfortunate since, as noted above, one doesn't always have an *a priori* guarantee that the target concept is in the concept class.

4 Discrete VC problem

In this section we define the computational problem of computing the VC dimension of a finite family of concepts defined over a finite domain, and give some simple results about its complexity. We then give a combinatorial characterization of such families when the VC dimension is 1.

4.1 Definition of the problem

We represent a concept class $\mathcal{C}$, $|\mathcal{C}| = r$, over a finite domain $\mathcal{X}$, $|\mathcal{X}| = n$, by an $r \times n$ matrix M such that $M_{i,j} = 1$ iff $x_j \in C_i$. Each row of M represents a concept in $\mathcal{C}$, and each column represents a point in $\mathcal{X}$. We define the VC dimension of the matrix M to be the VC dimension of the concept class represented by M.

Definition 4.1 *The* discrete VC dimension problem *is the following: given an $r \times n$ 0-1 valued matrix, M, to determine the VC dimension of M.*

To determine if the VC dimension of an $r \times n$ matrix M is less than d takes time $O(rn^d)$; it suffices to check all $\binom{n}{d}$ possible combinations of d columns.

Theorem 4.1 *The VC dimension problem can be solved in time $O(rn^{\lg r})$.*

Proof: Use the above argument and the fact that the VC dimension of a concept class with r concepts is at most $\lg r$. □

4.2 A combinatorial characterization

We now present a combinatorial characterization of matrices whose VC dimension is one.

Theorem 4.2 *A 0-1 valued matrix M has VC dimension* 1 *iff it can be reduced to the empty matrix by a sequence of operations of one of the following two forms:*

1. **Delete-column:** *Delete any column which contains less than two zeros or less than two ones.*
2. **Delete-row:** *Delete any row which is identical with a previous row in the matrix.*

Proof: The **delete-row** has the effect of removing a concept which is already represented by some other row in the matrix, so this operation can not affect the VC dimension of the matrix.

If M has VC dimension at least 2, then M has at least two columns and four rows such that all rows of the the induced submatrix are distinct. Thus those two columns will never be deleted by a **delete-column** operation. Therefore if the VC dimension is at least 2, then it remains so after a **delete-column** operation.

It remains to show that if neither **delete-row** nor **delete-column** is applicable, then M has VC dimension at least two.

Define the *score* of a column to be the maximum of the number of zeros it contains and the number of ones it contains. Let k be a column of maximum score, and assume without loss of generality that column k has at least as many ones as zeros. (See Figure 1.) Column k must have at least two zeros, since **delete-column** is not applicable. Let i_0 and i_1 be two rows in which column k has zeros, and let j be some column in which i_0 and i_1 have different values. (Column j must exist because **delete-row** does not apply.)

Now column j can not have more 1's than column k, so there is a row (say l_0) where column k contain has a 1 and column j has a 0. But at the same time, it is impossible that $M_{lj} = 0$ whenever $M_{lk} = 1$, for then the score of column j exceeds the score of column k. Therefore there is a row l_1 where $M_{l_1k} = M_{l_1j} = 1$. But then columns j and k are labelled in all possible ways by concepts i_0, i_1, l_0, and l_1, so that the VC dimension of M is at least two. □

$$\begin{array}{c|cc|}
 & k & j \\
\hline
i_0 & 0 & 0 \\
i_1 & 0 & 1 \\
 & 1 & \\
 & \vdots & \\
l_0 & 1 & 0 \\
 & & \\
l_1 & 1 & 1 \\
\hline
\end{array}$$

Figure 1: The matrix for the proof of Theorem 4.2

Now in order for the score of row j not to exceed the score of column k, in those rows where column k has ones column j must have at least one zero and at least one one. (Say in rows l_0 and l_1.) But then columns k and j represent two elements which can be labeled in all possible ways by concepts represented by the rows, so that the VC dimension of M is at least two. □

It remains an open problem to find a combinatorial characterization for VC dimensions greater than one, and to find a more efficient algorithm for the discrete VC dimension problem.

5 Conclusions

Our main result is an extension of distribution-free learning model to the case of infinite VC dimension; this greatly enlarges the class of concept classes which are pac-learnable. We have examined the closely related problem of approximating from examples, and have seen that our results probably do not generalize to this problem. Finally, we have considered the problem of computing the VC dimension for finite concept classes on finite domains, and have provided a new combinatorial characterization of the case that the VC dimension is one. A number of open problems, conjectures, and new research directions have been proposed.

Acknowledgements

We acknowledge many useful conversations with Michael Ben-Or, Gyora Benedek, Mauricio Karchmer, Manfred Warmuth and Michael Werman. The notions in section 3 were developed in response to a question raised by Manfred Warmuth. We are thankful for his permission to present the material here. We also wish to thank David Haussler for helpful comments on an earlier draft of this paper.

References

[1] G. M. Benedek. Ph.d. dissertation, in preparation. 1988.

[2] G. M. Benedek and A. Itai. Nonuniform learnability. In *ICALP*, pages 82–92, July 1988.

[3] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Classifying learnable geometric concepts with the Vapnik-Chervonenkis dimension. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, pages 273–282, Berkeley, California, May 1986.

[4] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. *Learnability and the Vapnik-Chervonenkis Dimension.* Technical Report USCS-CRL-87-20, U.C. Santa Cruz Computer Science Laboratory, Nov. 1987.

[5] A. Ehrenfeucht, D. Haussler, M. Kearns, and L. Valiant. *A General Lower Bound on the Number of Examples Needed for Learning.* Technical Report UCSC-CRL-87-26, U.C. Santa Cruz Computer Science Department, 1987.

[6] D. Haussler, M. Kearns, N. Littlestone, and M. K. Warmuth. Equivalence of models for polynomial learnability. In *First Workshop on Computational Learning Theory*, Aug. 1988. To be published by Morgan-Kaufmann.

[7] M. Kearns, M. Li, L. Pitt, and L. Valiant. On the learnability of boolean formulae. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 285–295, New York, New York, May 1987.

[8] M. Kearns, M. Li, L. Pitt, and L. Valiant. Recent results on boolean concept learning. In *Proceedings of the Fourth International Workshop on Machine Learning*, pages 337–352, University of California, Irvine, June 1987.

[9] D. Pollard. *Convergence of Stochastic Processes.* Springer-Verlag, 1984.

[10] J. R. Quinlan and R. L. Rivest. Inferring decision trees using the minimum description length principle. *Information and Computation*, 1988. (To appear. An early version appeared as MIT LCS Technical report MIT/LCS/TM-339 (September 1987).).

[11] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

[12] R. L. Rivest. Learning decision lists. *Machine Learning*, 2(3):229–246, 1987.

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

[14] V. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its applications*, XVI(2):264–280, 1971.