To appear in *Learning and Geometry*, David Kueker and Carl Smith, eds., Birkhauser.

Pac Learning, Noise, and Geometry

Robert H. Sloan*
Department of EE & Computer Science
851 S. Morgan St. Room 1120
University of Illinois at Chicago
Chicago, IL 60607-7053 USA

Abstract

This paper describes the probably approximately correct model of concept learning, paying special attention to the case where instances are points in Euclidean n-space. The problem of learning from noisy training data is also studied.

*Supported by NSF grant CCR-9108753. Email: `sloan@eecs.uic.edu`

1 Introduction

We will be concerned with that part of Machine Learning referred to as *concept learning* or *learning from examples*. The basic paradigm is as follows.

1.1 Example: An afternoon stroll

It is a bright, sunny afternoon. I take my robot out for a walk with me. We stop and sit down on a bench in a busy part of town. For every person who walks by us I say to my robot either, "That's a male," or "That's not a male," as the case may be. We stay quite some time, and eventually go home.

The following afternoon the weather is the same, and we go back to the same bench. This time, for every person who walks by, my robot says to me either "That's a male," or "That's not a male." My robot is correct ninety-six percent of the time.

1.2 The basic problem

The robot in the above fable (description of future research plans?) has come up with some rule for splitting the entire universe under consideration into two classes. That is a particular case of the following general class of problems.

There is some universe of objects under consideration, called the *instance space*. Frequently studied instance spaces include Euclidean n-space, bit vectors of some fixed length, and the integers. Of course, the set of all people is also a perfectly good instance space.

A *concept* is a subset of the instance space. An *example* of concept c consists of an instance, x, together with a label $c(x)$ that is 1 if $x \in c$ and 0 otherwise. (Throughout we will refer to concepts interchangeably as sets and $\{0,1\}$-valued functions.) If $c(x) = 1$ then x is a *positive instance* of concept c; otherwise x is a *negative instance*. The particular concept we are trying to learn is usually called the *target concept*. We always assume that the learner knows a priori that the target concept comes from particular *concept class*, or set of concepts. This is not a restriction, since the concept class could simply be the power set of the instance space.

Our learner is given a set of examples, or *sample* of the target concept, and asked to output the target concept (or some good approximation to it). In the example above, people constitute the instance space,[1] and the target concept is the set of male people. The robot demonstrates that it has found a good approximation to the target concept by providing a fairly accurate classification of previously unseen instances (people).

For the problems considered in this paper, that ability to classify is the point of the learning process. The examples given to the learner should be thought of as a training phase, so that in a future performance phase the learner can accurately classify instances for itself.

This problem has been studied by the Artificial Intelligence (AI) community since the very beginning of that field—primarily in an empirical fashion. More recently, the theoretical computer science community has taken an interest in this field. The work by this community began with Gold [11] back in the 1960s. (See Angluin and Smith [3] for an excellent survey of work through the early 1980s.) A seminal paper by Valiant [26] in 1984 prompted intensified effort. The goal of the work by the theory community, not surprisingly, is to provide precise models of the problem, and good algorithms to solve the problems so modeled (or proofs of the hardness of those problems).

The rest of this paper is organized as follows. The next section explains the method that theoretical computer scientists use for measuring problem difficulty. Then we will give a detailed definition of Valiant's model of concept learning (pac learning), and discuss why it is an attractive model. The remainder of the paper will concentrate on algorithms for that model, beginning briefly with finite instance spaces, and then concentrating on

Geometry the case where the instance space is Euclidean n-space, and the target concept is some region, and

Noise what happens to these algorithms when the examples used in the training phase are corrupted by noise.

[1] Perhaps it would be more accurate to say that the instance space consisted of the cross product of some set of discrete and continuous attributes (including, say, height, width, clothing type, etc.) that the robot actually observed.

2 Asymptotics

The particular interest of theoretical computer scientists studying concept learning, as opposed to say, statisticians, is finding *computationally efficient* learning algorithms. We will be concerned with two different measurements of efficiency:

1. The number of training examples needed.
2. The amount of computing time needed.

The standard definition of "efficient" consumption of whatever resource is under consideration is some amount polynomial in the "size of the problem."

For that to have any meaning, however, we must discuss, of course, the definition of problem size, but also the asymptotic difficulty of our problems. If we just consider any one fixed problem, it always has a "constant time" solution. For instance, the infamous Traveling Salesman problem for a fixed graph of 1000 nodes can be solved in constant time. (Of course, the "constant" in that case might be something around 2^{1000}.)

Thus instead of examining algorithms for one fixed learning problem, we always examine algorithms for an infinite sequence of learning problems. There are two different ways we can obtain such a sequence—by allowing the size/dimension of the instance space to grow, or by allowing the complexity of the concepts we consider to grow. We will now describe both these approaches briefly, but will concentrate on the first approach for the remainder of this paper.

2.1 Efficiency with respect to instance space dimension

Using this method, we parameterize the instance space by some natural dimension. Thus our instance space $X = \cup_{n=1}^{\infty} X_n$. For the sake of concreteness, in the rest of this paper we will always assume that our instance space X_n is one of E^n (Euclidean n-space), $[0,1]^n$, or $\{0,1\}^n$. We set $\mathbf{C} = \{(X_n, \mathcal{C}_n)\}_{n \geq 1}$, where $\mathcal{C}_n \subseteq 2^{X_n}$. We will use the term concept class to refer both to $\mathbf{C}$ and a particular $\mathcal{C}_n$. Note for future reference that when the literature refers to finite concept classes (or instance spaces), it means concept classes where $|\mathcal{C}_n|$ (or $|X_n|$) is finite for each n.

We are interested only in computationally efficient learning. Therefore, in addition to other assumptions, we always assume that our concept class has the property that there is a polynomial (in n) time algorithm for determining whether $x \in c$, for any $x \in X_n$ and $c \in \mathcal{C}_n$.

2.2 Efficiency with respect to concept complexity

The other way to make our learning problems asymptotic is to take a fixed instance space, but to allow concepts to grow arbitrarily complex. For instance, we could fix the instance space to be the real plane, and let the entire concept class be the set of all convex polygons. We could then make $\mathcal{C}_n$ the set of convex n-gons, or alternatively, the set of convex polygons with at most n edges. This approach to pac learning is explored by Linial, Mansour and Rivest, and by Blumer et al. [8, 19].

3 Pac learning

In this section we define Valiant's model of concept learning [26]. Angluin dubbed this model "probably approximately correct learning," [1] and it is thus referred to as *pac learning.*

In short, an algorithm pac learns from examples if it can, in a feasible amount of time, find (with high probability), a rule that is highly accurate. Now we must define what we mean by such terms as "find a rule," "with high probability," and "highly accurate."

For the purposes of this discussion, fix a concept class $\mathbf{C} = \{(X_n, \mathcal{C}_n)\}_{n \geq 1}$. First we define formally what it means for one concept in $\mathcal{C}_n$ to be an accurate approximation of the target concept (or any other concept, for that matter), and then we define pac learning itself.

DEFINITION. Fix n, and a probability distribution $\mathbf{P}$ on X. For any $0 \leq \epsilon \leq 1$, we say that concept c_1 is an ϵ-*approximation* of concept c_0 if and only if

$$\sum_{c_1(x) \neq c_0(x)} \mathbf{P}(x) \leq \epsilon. \tag{1}$$

The goal of our learning algorithm will be to find an ϵ-approximation of the target concept.

We assume that our learning algorithm has available to it a black box called EXAMPLES, and that each call to the black box returns a labeled example, (x, s), where $x \in X_n$ is an instance, and s is either 1 or 0 according to whether x is a positive or negative instance of the target concept. Furthermore, the EXAMPLES box generates the instances x according to some fixed probability distribution $\mathbf{P}$ on X_n. We make no assumptions whatsoever about the nature of $\mathbf{P}$, and our learner is not told what $\mathbf{P}$ is.

DEFINITION. Let $\mathbf{C} = \{(X_n, \mathcal{C}_n)\}_{n\geq 1}$ be a concept class. Algorithm A *probably approximately correctly learns (pac learns)* $\mathbf{C}$ if and only if for every $n \geq 1$, for every $c \in \mathcal{C}_n$, for every probability distribution on X_n, for every $0 < \epsilon, \delta \leq 1$, Algorithm A, given only n, ϵ, δ and access to EXAMPLES(c), meets the following two criteria.

Learning criterion Algorithm A outputs some representation of a concept c' such that

$$\Pr\left[c' \text{ is an } \epsilon\text{-approximation of } c\right] \geq 1 - \delta \qquad (2)$$

where the probability is taken over the output of EXAMPLES and any coin tosses A may make.

Efficiency criterion The running time of A is bounded by some polynomial function of $1/\epsilon$, $1/\delta$, and n.

We say that a concept class $\mathbf{C}$ is *pac learnable* if there exists some algorithm that pac learns $\mathbf{C}$.

We note here for later use the following weaker definition.

DEFINITION. *Statistical pac learning or s-pac learning* has the same definition as pac learning except that the efficiency criterion requires the sample complexity, rather than the running time, to be bounded by by some polynomial function of $1/\epsilon$, $1/\delta$, and n.[2]

4 Discussion of the definition

Intuitively, we are saying that the learner is supposed to do the following:

[2]Terminology in this field is not yet completely settled. Many authors use the term pac learning for what we are calling statistical pac learning, and polynomial pac learning for what we are calling pac learning.

1. Ask nature for a random set of examples of the target concept.

2. Run in polynomial time.

3. Output some concept that with high probability agrees with the target concept on most of the instances.

We think of Nature as providing examples to the learner according to the (unknown) probability with which the examples occur in Nature. Though the learner does not know this probability distribution, he does know that the concept he outputs needs to closely approximate the target concept for only this probability distribution.

One of the great strengths of pac model is that it places no restrictions whatsoever on nature of the probability distribution on the instance space. For this reason, the other common name for this model besides pac learning is *the distribution-free model* of (concept) learning.

The two probabilistic quantifiers in the definition are necessary to make the problem possible to solve. Intuitively, there may be some extremely bizarre but low probability examples that occur in Nature, and it would be unreasonable to demand that the learner's output concept classify them correctly. Hence we require only approximate correctness. Moreover, since the examples the learner receives from Nature are drawn stochastically, there is some small but nonzero chance that those examples will be wildly unrepresentative. Therefore we cannot demand that the learner *always* output an approximately correct concept; we require that the learner do so only with high probability.

The notion of statistical pac learning was studied in the statistical pattern recognition literature before the notion of pac learning was introduced in 1984. The difference between the two is that statistical pac learning requires us to be efficient in our consumption of examples but not in our computation time. Notice that any algorithm that pac learns must statistically pac learn since the sample complexity of an algorithm is a lower bound on its running time. Thus the strongest infeasibility results we can obtain are showing the infeasibility of statistical pac learning. In general, it is often easier to prove theorems about statistical pac learning than about pac learning.

The excellent survey article of Kearns et al. gives a more lengthy discussion of the entire pac learning model, and also provides an overview of early results obtained using this model [18]. Blumer et al. and Valiant both contain additional material on the motivation for the pac learning model [8, 26].

5 Variations on the definition of pac learning

The pac learning model can be altered in many ways. For instance, instead of having a single probability distribution on the whole instance space and a single box EXAMPLES, there can be two separate probability distributions, one each for positive examples and negative examples, and two boxes, EXAMPLES^+ and EXAMPLES^-, each giving only one kind of instance. In this case, the learning algorithm is allowed to make calls to either box, and its hypothesis is required to come within ϵ on both positive and negative examples. This model is called the two-button model; the model we are using is called the one-button model.

There are in fact numerous different minor variations on pac learnability. Haussler et al. compare many of these variations, and show them to be substantially equivalent [13].

6 An example of pac learning

We now exhibit an algorithm that pac learns monomials [26]. The instance space X_n is $\{0,1\}^n$ for some positive n. The concept class is the set of all boolean expressions in n variables that can be represented as monomials (simple conjunctions of literals).

Algorithm A for pac learning is as follows: Algorithm A initializes its hypothesis to be the conjunction of all $2n$ literals, $x_1\bar{x}_1 \cdots x_n\bar{x}_n$. Algorithm A then makes m calls to EXAMPLES, where

$$m = \frac{1}{\epsilon}\left(n \ln 3 + \ln(1/\delta)\right).$$

For each positive example, $(instance, 1)$, Algorithm A crosses off x_i from its hypothesis if x_i is 1 in $instance$, and crosses off $\bar{x}_i$ if x_i is 0 in $instance$. Algorithm A simply ignores negative instances. The output of A is the monomial consisting of whatever literals have not been crossed off after the m

instances are seen. (If all of the m instances are negative, then A's hypothesis is $x_1\bar{x}_1 \cdots x_n\bar{x}_n$, which is always false.)

The running time of of A is clearly polynomial in $1/\epsilon, 1/\delta$, and n, so A meets the efficiency criterion for pac learning. In Section 7 we will give a general result that proves that Algorithm A meets the learning criterion for pac learning. (In fact, a somewhat better bound for m can be obtained by making a direct combinatorial argument.)

Note for later reference that the correctness of Algorithm A is critically dependent on the data from EXAMPLES being noise free. One negative instance, mistakenly labeled positive, would most likely cause the algorithm to output some totally bogus hypothesis.

7 Pac learning with finite concept classes

Let us briefly examine sufficient conditions for a concept class to be pac learnable. A starting place place is any concept class of very small cardinality. In the extreme, consider the case of a concept class containing only two concepts, c_1 and c_2.

Recall that we always assume it is computationally efficient to determine whether any particular instance is contained in a given concept. Now all we need to do is draw a sufficiently large sample from EXAMPLES, and select whichever concept is consistent with the sample, or, if both concepts are consistent, choose either one arbitrarily.

We can extend this line of reasoning to considerably larger (finite) concept classes. Our treatment follows Blumer et al. [7]. Consider an algorithm that draws a sample of size $m(|\mathcal{C}|, \epsilon, \delta)$, and then outputs any concept $c \in \mathcal{C}$ that is consistent with the sample. Ignoring for a moment the issue of computation time, what is a lower bound on the function m that insures that the learning algorithm meets the statistical pac learning condition?

DEFINITION. Fix a target concept and a probability distribution on the instance space. We say that a concept is *ϵ-good* if it is an ϵ-approximation of the target concept, and *ϵ-bad* if it is not.

We want to guarantee that the probability that the algorithm returns an ϵ-bad concept is at most δ. The probability that a particular ϵ-bad concept would be consistent with m randomly drawn examples is bounded above by

$$(1-\epsilon)^m.$$

A crude upper bound on the probability that *any* ϵ-bad concept is consistent with m randomly drawn examples is

$$|\mathcal{C}|\,(1-\epsilon)^m.$$

Hence we want to choose m large enough to insure that

$$|\mathcal{C}|\,(1-\epsilon)^m \leq \delta. \tag{3}$$

Taking logs of both sides, and solving for m yields

$$m \geq \frac{1}{-\ln(1-\epsilon)}\left(\ln(|\mathcal{C}|) + \ln(1/\delta)\right). \tag{4}$$

We can write a simpler inequality that implies inequality (4) by noticing that by Taylor's Theorem

$$\begin{aligned} \ln\left(\frac{1}{1-\epsilon}\right) &> \ln 1 + \epsilon\frac{1}{(1-\epsilon)} + \frac{\epsilon^2}{2!}\frac{1}{(1-\epsilon)^2} \\ &> \frac{\epsilon}{1-\epsilon} \\ &> \epsilon. \end{aligned}$$

Hence, $1/\epsilon > 1/-\ln(1-\epsilon)$, and we have shown that

Theorem 1 *Fix a concept class* $\mathbf{C} = \{(X_n, \mathcal{C}_n)\}_{n\geq 1}$. *Any algorithm which on input* n, ϵ, *and* δ *makes*

$$m = \frac{1}{\epsilon}\left(\ln(|\mathcal{C}_n|) + \ln\left(\frac{1}{\delta}\right)\right). \tag{5}$$

calls to EXAMPLES *and returns any concept consistent with that sample statistically pac learns* $\mathbf{C}$.

We say that m is an upper bound on the *sample complexity* of the concept class $\mathbf{C}$. More generally we say that an arbitrary concept class $\mathbf{C}$ has sample complexity at most m if there is a learning algorithm with sample complexity m that statistically pac learns $\mathbf{C}$.

As long as $|\mathcal{C}_n| = O\left(2^{p(n)}\right)$, for some polynomial p, then $\mathbf{C}$ has polynomial sample complexity, and is statistically pac learnable. (If $|\mathcal{C}_n| = O\left(q(n)\right)$, for some polynomial q, then $\mathbf{C}$ is trivially pac learnable.) If $\mathbf{C}$ has polynomial sample complexity, and if the problem of finding a concept from $\mathbf{C}$ consistent with a given sample can be solved in polynomial time, then $\mathbf{C}$ is pac learnable.

There are 3^n different monomials on n boolean variables, so the concept class of monomials clearly has polynomial sample complexity. On the other hand, we obviously cannot exhaustively examine all 3^n monomials in polynomial time. The algorithm for learning monomials given above in Section 6 overcomes this problem by finding a consistent concept in a more clever manner.

8 Remark on the form of the sample complexity

The general form of Equation (5),

$$\frac{1}{\epsilon}\ln\left(\frac{1}{\delta}\right)+\frac{1}{\epsilon}\left(\text{term without } \delta \text{ concerning concept class}\right),$$

is one that crops up time and again in formulas for sample complexity. In this section we will attempt to see why this is so, and especially why the dependence on δ is almost always of the form $(1/\epsilon)\ln(1/\delta)$.

The explanation begins with the fact that Equation (5) is indeed a choice that makes the inequality

$$(1-\epsilon)^m \leq \delta \tag{6}$$

true. It is easy to see that Inequality (6) must always be true. For any concept class $\mathbf{C} = \{(X_n, \mathcal{C}_n)\}_{n\geq 1}$, we can always add some new distinguished element x to every instance space, and for every concept c in the class add a new concept $c \cup \{x\}$. If probability weight ϵ is placed on x, then our sample must always contain x, because the output concept must classify x correctly to meet the learning criterion, and there is no way to know anything about the classification of x without actually seeing its label. Hence we have a clear lower bound on sample complexity of $(1/\epsilon)\ln(1/\delta)$.

Can we push this line of reasoning farther, and get a larger lower bound than $(1/\epsilon)\ln(1/\delta)$? The most extreme form of this idea would be to design an instance space of size $n = 1/\epsilon$, put probability weight ϵ on each instance, and make the concept class the full power set of the instance space. Now we must draw enough examples to insure that we see all n instances at least once with probability at least $1-\delta$. This is the well known coupon collector's problem.[3] It turns out that we have the convergence

$$\Pr\left[cn + n \ln n \text{ draws are sufficient}\right] \rightarrow e^{-e^{-c}}. \tag{7}$$

(In fact, the convergence is quite rapid.)

Choosing $c = \ln(1/\delta)$ makes the right-hand side of Equation (7) equal to $e^{-\delta}$, which is approximately $1-\delta$, since for a of small absolute value we have $e^a \approx 1 + a$. Thus even in this case we still need to draw only

$$\frac{1}{\epsilon} \ln\left(\frac{1}{\delta}\right) + \frac{1}{\epsilon} \ln\left(\frac{1}{\epsilon}\right)$$

examples to have a $1-\delta$ probability of success.

9 Continuous instance spaces: the VC dimension

Consider now a typical geometric learning problem, $\mathbf{C} = (E^n, B^n)$, where B^n represents the set of all axis-parallel boxes contained in E^n. Now our concept class has uncountable cardinality, so the method of Theorem 1 cannot work, nor can any method based on the size of the concept class.

It turns out, however, that a combinatorial parameter of set systems can take the place of the cardinality of the concept class. In fact, this parameter, the Vapnik-Chervonenkis dimension [29], hereinafter VC dimension (or in symbols $\mathrm{VCD}(\cdot)$), provides both a necessary and sufficient condition for the existence of an algorithm that can statistically pac learn just by drawing a fixed sample size. Since the VC dimension is a combinatorial parameter of set systems, we define it in terms of sets, though of course the sets we are interested are concepts.

9.1 VC dimension definition

DEFINITION. Fix a domain X, and let $\mathcal{C} \subseteq 2^X$. A set $S \subset X$ is *shattered* (by $\mathcal{C}$) if for each subset $S' \subseteq S$, there is a set $c \in \mathcal{C}$ which contains all of S', but none of the points in $S - S'$.

[3] So called because another way of stating it is, "A cereal manufacturer is putting n distinct coupons into cereal boxes with equal probability. How many boxes of cereal must one buy to insure getting at least one of each coupon?"

Remark: The term shatter is now well established. However, as Pollard points out [22], the right picture to keep in mind is not really S being broken into lots of tiny pieces by $\mathcal{C}$. Rather, one should imagine a diligent $\mathcal{C}$ picking out all the different subsets of S.

DEFINITION. The *VC dimension* of $\mathcal{C} \subseteq 2^X$ is the cardinality of the largest set of points from X shattered by $\mathcal{C}$.

Examples: The class of all rectangular regions in the plane has VC dimension 4. The class B^n on E^n has VC dimension $2n$. The class of all spheres in E^n has dimension $n+1$. For any finite concept class $\mathcal{C}$, we have $VCdim(\mathcal{C}) \leq \log(|\mathcal{C}|)$.[§] There are now a number of papers which calculate the VC dimension of different classes [4, 9, 14, 16].

The VC dimension was first studied in connection with statistical pattern recognition. Pollard and Vapnik have both written good books discussing it from that point of view [22, 28]. The first source that I am aware of to point out that it has some connection to efficient concept learning is Pearl [20].

9.2 Algorithms

For our purposes, the key fact about the VC dimension is that a fixed concept class has polynomial sample complexity if and only if it has finite VC dimension, or a sequence of concept classes $\mathcal{C}_n$ has polynomial sample complexity if and only if $\mathrm{VCD}(\mathcal{C}_n)$ is polynomial in n. In particular, Blumer et al. showed [8]

Theorem 2 *For any instance space X and "well-behaved" $C \subseteq 2^X$, any learning algorithm which obtains a sample of size*

$$m \geq \max\left(\frac{4}{\epsilon}\log\left(\frac{2}{\delta}\right), \frac{8\mathrm{VCD}(C)}{\epsilon}\log\left(\frac{13}{\epsilon}\right)\right) \tag{8}$$

and returns any concept consistent with all m examples meets the learning criterion of the pac learning definition.

Technical note on "well-behaved" classes: For anything to work with geometric instances spaces, we must have certain conditions which amount to saying that the concepts are not measure-theoretically pathological. For instance, we certainly require that every concept is a Borel set. A fair amount

[§] Unless explicitly stated otherwise, all logarithms in this work are base 2.

has been written about this issue [5, 6, 8]. In practice, this is a nonissue. As computer scientists, we are really interested only in *exceedingly* well-behaved concept classes. After all, our end goal is to have all these problems analyzed by real-world finite-precision computers. (There may be algorithms where we have to be concerned with round-off errors and numerical stability, but that is a completely separate issue.)

Corollary 1 *Let* $\mathbf{C} = \{(X_n, \mathcal{C}_n)\}_{n\geq 1}$ *be a concept class such that* $\mathrm{VCD}(\mathcal{C}_n) = p(n)$ *for some polynomial* p. *Then* $\mathbf{C}$ *is statistically pac learnable, with a sample size of*

$$m \geq \max\left(\frac{4}{\epsilon}\log\left(\frac{2}{\delta}\right), \frac{8p(n)}{\epsilon}\log\left(\frac{13}{\epsilon}\right)\right). \tag{9}$$

To go from statistical pac learning to true pac learning, it is sufficient to be able to find a concept consistent with a sample in polynomial time. For instance, for the class of axis-parallel boxes mentioned above, this is a simple task. Simply take a sample of the size specified by Corollary 1, and return the smallest axis-parallel boxes containing all the positive instances seen, which can easily be done in time linear in the number of points.

Much of the labor of pac learning researchers consists of searching for clever ways to find a concept consistent with a sample. Unfortunately, for many interesting classes, finding a concept consistent with a sample is NP-complete (which most computer scientists believe means that it is impossible to find a polynomial-time algorithm for this problem). Theorem 2 can be strengthened somewhat to show that with a slightly larger sample size it is sufficient to find a concept that is almost consistent with a sample. In theory, this sometimes makes the problem of finding an acceptable concept much easier computationally, although in practice developing algorithms to find almost consistent concepts is typically still very difficult.

9.3 Lower bounds on sample complexity

The VC dimension also gives us a sort of lower bound on sample complexity for pac learning [10].

Theorem 3 *Fix a concept class* $\mathbf{C} = \{(X_n, \mathcal{C}_n)\}_{n\geq 1}$, *such that* $|\mathcal{C}_n| \geq 3$ *for sufficiently large* n. *Consider any (possibly randomized) algorithm that works* *by calling* EXAMPLES $m = f(\epsilon, \delta, n)$ *times for some* f*, and returning any concept consistent with those* m *examples. To meet the pac learning criterion, it must be that*

$$m = \Omega\left(\frac{1}{\epsilon}\ln\left(\frac{1}{\delta}\right) + \frac{\mathrm{VCD}(\mathcal{C}_n)}{\epsilon}\right).$$

Theorem 3 would seem to imply that we must always have a finite VC dimension in order to pac learn. However, this is not strictly true. The limitations imposed by Theorem 3 can be overcome by algorithms with slightly different behavior. For instance, the class of all finite unions of closed intervals in $[0, 1]$ has infinite VC dimension, but does have a pac learning algorithm [19].[5] The algorithm behaves roughly as follows. It first draws a small sample from EXAMPLES, and makes as its first hypothesis some "small" concept (one that is a union of as few intervals as possible) consistent with that sample. Then another sample is drawn from EXAMPLES, and the initial hypothesis is checked for consistency with it. If the initial hypothesis is not consistent that sample, then the algorithm starts over again, but drawing more examples. This process is repeated as many times as necessary until a hypothesis is found that is consistent with a subsequently drawn sample.[6]

10 Noisy data

So far, we have assumed that all our training data provided by the box EXAMPLES is perfect. In the real world, of course, there is always some noise.

Noise is modeled by replacing the box EXAMPLES with some other box which provides noisy examples. In theoretical computer science jargon, such boxes are usually called "oracles." One can think of these other oracles as first getting an example from EXAMPLES, and then adding noise to it. The output of the noise oracles is all that the learner can observe.

[5] This is an example of using concept complexity, rather than instance space dimension, as the measure of problem size, as was discussed in Section 2.2.

[6] Technically, this algorithm does not meet our definition of pac learning, since it obviously does not always halt in polynomial time. It does, however, have halt in expected polynomial time. See Haussler et al. [13] for discussion about what difference this small perturbation in the definition of pac learning might make.

Three oracles have been studied so far that are relevant to geometric instance spaces (as have several others that apply only to the instance space $\{0,1\}^n$ [12, 24]). The "desired," noiseless output of each oracle is a correctly labeled example (x, s), where x is drawn according to $\mathbf{P}$. We now now give a precise description of the actual outputs from the following noise oracles: MAL_ν [27], RMC_ν [2], MMC_ν [24, 25].

- When MAL_ν is called, with probability $1-\nu$, it does indeed return a correctly labeled (x, s) where x is drawn according to $\mathbf{P}$. With probability ν it returns an example (x, s) about which no assumptions whatsoever may be made. In particular, this example may be maliciously selected by an adversary who has infinite computing power, and has knowledge of the target concept, $\mathbf{P}$, ν, and the internal state of the algorithm calling this oracle. This *malicious noise* oracle models the situation where the learner usually gets a correct example, but some small fraction ν of the time the learner gets noisy examples, and the nature of the noise is unknown or unpredictable.

- When RMC_ν is called, it calls EX to obtain some (noiseless) (x, s), and with probability $1-\nu$, RMC_ν returns (x, s). However, with probability ν, RMC_ν returns $(x, \overline{s})$. This *random misclassification noise* oracle models a benign form of misclassification noise.

- When MMC_ν is called, it also calls EX to obtain some (noiseless) (x, s), and with probability $1-\nu$, MMC_ν returns (x, s). With probability ν, MMC_ν returns (x, l) where l is a label about which no assumption whatsoever may be made. As with MAL_ν we assume an omnipotent, omniscient adversary; but in the case the adversary gets to choose only the label of the example. This *malicious misclassification noise* oracle models a situation in which the only source of noise is misclassification, but the nature of the misclassification is unknown or unpredictable.

10.1 Malicious noise

Not surprisingly, giving an all powerful adversary the ability to tamper with our training data restricts our ability to learn. Kearns and Li [17] showed that, given certain very minimal assumptions, it is impossible to pac learn with examples from MAL_ν unless $\nu < (\epsilon/1+\epsilon)$. The proof is information-theoretic, meaning that it is impossible even to statistically pac learn under such circumstances.

A generalization of Theorem 2 shows that for very low noise rates, it is sufficient to draw a polynomial-size sample from MAL_ν and return any concept in the concept class that has a minimum disagreement rate with the sample. (The disagreement of a concept with respect to a sample is the number of instances classified differently by the sample and the oracle which provided the examples.)

There is, however, a better method. Any pac learning algorithm for a concept class **C** can be converted to be robust against a malicious noise with a noise rate up to

$$O\left(\frac{\epsilon}{\mathrm{VCD}(\mathbf{C})}\log\left(\frac{\mathrm{VCD}(\mathbf{C})}{\epsilon}\right)\right).$$

The key idea of this meta-algorithm [17] is as follows. The meta-algorithm obtains many, many examples from MAL_ν. These are broken up into blocks of length s, where s is the number of noise-free examples the regular pac learning algorithm requires for **C**. Each block is fed to the regular algorithm, and for each it makes a hypothesis. The number of blocks is chosen so that with high probability one of the blocks is noise-free. One final set of examples is drawn, and the meta-algorithm tests each hypothesis of the regular algorithm on this set, choosing the hypothesis with the lowest disagreement rate.

10.2 Labeling noise

Angluin and Laird studied the case where the only source of noise is random mislabeling [2]. They showed that in this case, for any noise rate strictly less than one half, it is sufficient to pick out the concept with minimum disagreement rate, or even any concept with low disagreement rate. This solves the problem of statistical pac learning. These results extend to the case where the label is maliciously chosen by a powerful adversary (i.e., where RMC_ν is replaced by MMC_ν) [24, 25].

Unfortunately, in this case no meta-algorithm for converting any pac learning algorithm into an algorithm robust against large amounts of mislabeling noise is known. Angluin and Laird do give a particularly clever algorithm for finding a low disagreement rate concept for the concept class

k-CNF (a particular class of boolean formulas on $\{0,1\}^n$).

11 Research directions

Perhaps the most important research goal right now should be to find more pac learning algorithms for more concept classes, concentrating on true pac learning, not statistical pac learning. The most notorious open class is arbitrary DNF formulas on $\{0,1\}^n$. Either an algorithm, or a prove that finding one is impossible (or NP-hard, or equivalent to factoring or what have you) would be of great interest. Of course, it would be nicer to discover a good algorithm than to discover a hardness proof.

Another important open class involves unions of boxes. Algorithms exist for learning arbitrary unions of axis-parallel rectangles in E^2, and arbitrary nested differences of axis-parallel boxes in E^n [8, 15]. Arbitrary unions of boxes (regardless of whether they are axis-parallel) in E^n cannot be pac learnable since the VC dimension is not polynomial in n. It is open whether unions of up to $O(n)$ axis-parallel boxes (or even of $O(\log n)$) in E^n can be pac learned.[7] This concept class is of particular interest because it is also sometimes used in artificial intelligence approaches to data classification (e.g., [23]).

I believe that in the end, the number of good polynomial-time pac learning algorithms found over the intermediate-term future will determine whether this whole line of research continues to flourish, or becomes just one more interesting little side area studied for some period of time by theoretical computer science.

Nevertheless, there is room for a lot of work in the area of noise and geometry. The malicious noise oracle certainly is sufficient to model any conceivable real-world noise, but it is so strong that, as we saw, only limited results are possible. Other than that model, none of the noise models studied to date seems particularly compelling for geometric domains. It would be interesting to study what happens if, say, labels are not affected at all by noise, and instances are perturbed by noise with a normal distribution.

[7]Note that this is related to the problem of DNF boolean formulas mentioned above. A union of n boxes in E^n would generalize learning n-term DNF in $\{0,1\}^n$. This problem is known to be NP-complete if we demand that the output be a union of a *minimum* number of boxes [21]

References

[1] Dana Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1987.

[2] Dana Angluin and Philip Laird. Learning from noisy examples. *Machine Learning*, 2(4):343–370, 1988.

[3] Dana Angluin and Carl H. Smith. Inductive inference: Theory and methods. *Computing Surveys*, 15(3):237–269, September 1983.

[4] Patrick Assouad. Densité et dimension. *Ann. Inst. Fourier, Grenoble*, 33(3):233–282, 1983.

[5] S. Ben-David, A. Itai, and E. Kushilevitz. Learning by distances. In *Proc. 3rd Annu. Workshop on Comput. Learning Theory*, pages 232–245, San Mateo, CA, 1990. Morgan Kaufmann.

[6] Shai Ben-David and Gyora M. Benedek. Measurability constraints on pac learnability. Technical report, Technion, Haifa, 1991.

[7] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Occam's razor. *Information Processing Letters*, 24:377–380, April 1987.

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

[9] R. M. Dudley. A course on empirical processes. In *Lecture Notes in Mathematics No. 1097*. Springer-Verlag, 1984.

[10] Andrzej Ehrenfeucht, David Haussler, Michael Kearns, and Leslie Valiant. A general lower bound on the number of examples needed for learning. *Information and Computation*, 82:247–261, 1989.

[11] E. Mark Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.

[12] Sally A. Goldman and Robert H. Sloan. Can PAC learning algorithms tolerate random attribute noise? *Algorithmica*, 1995. To appear. Preliminary version available as "The Difficulty of Random Attribute Noise," Washington University Dept. of Computer Science Tech. Report WUCS-91-92, 1991.

[13] David Haussler, Michael Kearns, Nick Littlestone, and Manfred K. Warmuth. Equivalence of models for polynomial learnability. *Information and Computation*, 95(2):129–161, December 1991.

[14] David Haussler and Elmo Welzl. Epsilon-nets and simplex range queries. *Discrete Computational Geometry*, 2:127–151, 1987.

[15] David Helmbold, Robert Sloan, and Manfred K. Warmuth. Learning nested differences of intersection-closed concept classes. *Machine Learning*, 5(2):165–196, 1990.

[16] David Helmbold, Robert Sloan, and Manfred K. Warmuth. Learning integer lattices. *SIAM Journal on Computing*, 21(2):240–266, 1992.

[17] M. Kearns and M. Li. Learning in the presence of malicious errors. *SIAM J. Comput.*, 22:807–837, 1993.

[18] Michael Kearns, Ming Li, Leonard Pitt, and Leslie 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.

[19] Nathan Linial, Yishay Mansour, and Ronald L. Rivest. Results on learnability and the Vapnik-Chervonenkis dimension. *Information and Computation*, 90(1):33–49, January 1991.

[20] Judea Pearl. On the connection between the complexity and credibility of inferred models. *Journal of General Systems*, 4:255–264, 1978.

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

[22] David Pollard. *Convergence of Stochastic Processes*. Springer-Verlag, 1984.

[23] Steven Salzberg. A nearest hyperrectangle learning method. *Machine Learning*, 6(3):251–276, 1991.

[24] Robert H. Sloan. Four types of noise in data for PAC learning. *Information Processing Letters*. To appear.

[25] Robert H. Sloan. Types of noise in data for concept learning. In *First Workshop on Computational Learning Theory*, pages 91–96. Morgan Kaufmann, 1988.

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

[27] Leslie G. Valiant. Learning disjunctions of conjunctions. In *Proceedings IJCAI-85*, pages 560–566. International Joint Committee for Artificial Intelligence, Morgan Kaufmann, August 1985.

[28] V. N. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, New York, 1982.

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