When Can Two Unsupervised Learners Achieve PAC Separation?

Paul W. Goldberg
Dept. of Computer Science,
University of Warwick,
Coventry CV4 7AL, U.K.
pwg@dcs.warwick.ac.uk

November 1, 2000

Abstract

In this paper we introduce a new framework for studying PAC learning problems, that has practical as well as theoretical motivations. In our framework the training examples are divided into the two sets associated with the two possible output labels, and each set is sent to a separate (unsupervised) learner. The two learners must independently fit probability distributions to their examples, and afterwards these distributions are combined to form a hypothesis by labeling test data according to maximum likelihood. That is, the output label for any input would be the one associated with the unsupervised learner that gave that input the higher probability.

This approach is motived by the observation that PAC learning algorithms of this kind are extendable in a natural way to multi-class classification. It also turns out to introduce new algorithmic challenges, even for very simple concept classes. Learning may be viewed as a cooperative game between two players, for which we must design a strategy that they will both use.

Within the framework, we give algorithms for learning various simple geometric concept classes. In the boolean domain we show how to learn parity functions, and functions for which there is a constant upper bound on the number of relevant attributes. We give an algorithm for learning monomials subject to the assumption that the input distribution is a product distribution. The main open problem is whether monomials (or any other concept class) distinguish learnability in this framework from standard PAC-learnability.

1 Introduction

In a standard learning problem in the Probably Approximately Correct (PAC) setting, there is a source of data consisting of instances generated by a probability distribution D over a domain X, labeled using an unknown function $f : X \longrightarrow \{0, 1\}$. Thus f divides members of X into two sets $f^{-1}(0)$ and $f^{-1}(1)$, and the learner's objective is to find good approximations to those sets. The main algorithmic challenge is usually to find any classification function h that agrees with the observed data, where h must be chosen from a class of functions whose diversity is limited. Thus learning is all about separating the two sets.

The above scenario is a special case of *supervised learning*, where the learner's task is to learn a function. Another learning task which arises frequently in practice is *unsupervised learning*, where the task is to estimate a probability distribution that generated a collection of observations, assumed to be sampled from that distribution. There are not so many papers on unsupervised learning in the learning theory literature; the topic was introduced in [13], see also [5, 7, 8, 6]. This paper is motivated by the simple observation that it may be possible to use unsupervised learning to do supervised learning, especially when the supervised learning problem is to learn a function whose range is a small set $V = \{v_1, \ldots, v_k\}$. The general approach is to fit probability distributions $D_1, \ldots, D_k$ to the sets $f^{-1}(v_1), \ldots, f^{-1}(v_k)$, and derive a classifier h which labels any $x \in X$ by the value v_i, where i maximizes $D_i(x)$ (*i.e.* maximizes the likelihood).

This approach to multi-class classification (for problems such as handwritten digit recognition) has several advantages over the separation approach:

1. All classes are treated in the same way. We avoid having to start by choosing some partition of the classes into two sets for the purpose of applying binary separation. Such a partition treats class labels lying on opposite sides differently from class labels lying on the same side.

2. For applications such as handwritten digit recognition, it is more natural to model the data generation process in terms of 10 separate probability distributions, than as a collection of thresholds between different digits.

3. The approach is robust to situations where class overlap occurs (as is usually the case in practice). Class overlap refers to a situation in a classification problem where an input may have more than one possible valid output (class label). Standard PAC algorithms do not address this problem (although there have been extensions such as "probabilistic concepts" [14] that do so). Moreover, if we have a model for how members of each separate class are generated, we can easily modify the derived classification function to deal with a variable misclassification penalty function, or changes in the class priors. This kind of modification cannot be applied to a simple dividing line between pairs of classes.

In practical studies such as [15] that build a multi-class classifier from a collection of 2-class classifiers, a distinction is made between separating each class from the union of the others (*1-v-r* classifiers, where 1-v-r stands for one-versus-rest) and pairwise separation (*1-v-1* classifiers). Neither are entirely satisfactory – for example it may be possible to perform linear 1-v-1 separation for all pairs of classes, but not linear 1-v-r separation, while a problem with 1-v-1 classification (as studied in [15]) is the difficulty of combining

the collection of pairwise classifiers to get an overall classification, in a principled way, for example ensuring that all classes are treated the same way. In [15], the first test for any unlabeled input is to apply the separator that distinguishes 0 from 9. Thus 0 and 9 are being treated differently from other digits (which in turn are also treated differently from each other.) See also [9] for a review of methods for using binary classifiers to construct a multi-class classifier.

Class overlap is the main practical reason why it is generally inappropriate, or just simply impossible, to find a separator that isolates inputs with one class label from those with another. If class overlap is really present but class separation is nevertheless achieved, it usually means the model has overfitted the data. In these situations, it is wrong to try to separate the different classes perfectly.

From a theoretical PAC learning perspective, we are addressing a natural question in asking to what extent it hampers the learner not to have simultaneous access to examples with different labels. The question raises novel algorithmic challenges, even for some of the simplest PAC learning problems. Indeed, a notable feature of this framework is the number of hard open problems that seem to arise. In contrast with all other studies of unsupervised learning, we are interested in its usefulness in fulfilling a given supervised learning objective, as opposed to approximating the probability distribution that generated the data. As we will see, hypothesis distributions need not be good approximators in order for their combination to achieve PAC learning.

The main theoretical question (which we leave open) is of course: are all PAC learnable problems also learnable in this framework, and if not, how does the set of learnable problems compare with other subsets of PAC learnable problems, for example Statistical Query (SQ) learnability [12]. (In the case of SQ learning, we find that parity functions are learnable in this framework but from [12] we know they are not learnable using SQs.)

1.1 Formalizing the Learning Framework

We use the standard convention of referring to the two classes of inputs associated with the two class labels as the "positive examples" and the "negative examples". Each learner has access to a source of examples having one of the two class labels. More precisely, one learner may (in unit time) draw a sample from D restricted to the positive examples, and the other may draw a sample from D restricted to the negative examples. This formalism loses any information about the class priors, *i.e.* the relative frequency of positive and negative examples, but PAC learnability is equivalent to PAC learnability where the relative frequency of positive/negative examples is concealed from the learner. (Formally, this is the equivalence of the standard PAC framework with the "two-button" version, where the learner has access to a "positive example oracle" and a "negative example oracle" [10]. The two-button version conceals the class priors and only gives the learner access to the distribution as restricted to each class label.)

We assume that neither unsupervised learner knows whether it is receiving the positive or the negative examples. For a concept class that is closed under complementation, such information would be of no help to the learners. For a concept class that is not closed under complementation (such as rectangles), note that it is PAC learnable if and only if its closure under complementation is PAC learnable. This means that any learning algorithm in our framework which required class labels to be provided to each of the two learners, would lack a robustness property that standard PAC algorithms have. Regarding

our main open question of whether there exists a concept class that is PAC-learnable but is not learnable in our framework, any concept class that distinguishes PAC learnability from learnability with unsupervised learners that received labeled examples, would have a closure under complementation that would distinguish PAC learnability from learnability with unsupervised learners that receive *un*labeled examples. Hence we assume examples are unlabeled.

Note however that we find that for some concept classes that are not closed under complementation (notably monomials, section 3.2) we can (without much difficulty) find algorithms in our setting that use labeled examples, but we have so far not found any algorithm that works with unlabeled examples.

The class label which the pair of learners assign to input x is the one associated with the data sent to the learner that assigned x the higher likelihood. If x is given the same likelihood by both distributions generated by the learners, the tie is broken at random. (This kind of tie may occur when both distributions give a probability of 0 to x, a situation where "conservative" hypotheses are appropriate).

1.2 Notation and Terminology

We refer to the two unsupervised learners as learner A and learner B, and we also assume by convention that learner A is the one receiving the positive examples, however as we have noted above, learner A (and likewise learner B) is not told its identity.

Theorem 1 below justifies the design of algorithms in which instead of insisting that the outputs of the unsupervised learners define probability distributions, we allow the outputs to be any real values. We say that each learner must construct a *scoring function* from the domain X to the real numbers $\mathbf{R}$. The comparison of the two scores assigned to any $x \in X$ is used to determine the class label that the hypothesis classifier assigns to x.

We will say that learner A (respectively B) "claims" an input x if it gives x a higher likelihood or score than learner B (respectively A). We say that a learner "rejects" an input if it assigns a likelihood of 0, or alternatively a score of minimal value (it is convenient to use $-\infty$ to denote such a score). Thus if a learner rejects an input, it will be claimed by the other learner provided that the other learner does not also reject that input.

2 General Results

In this section we give some general results about the two unsupervised learners framework. (Then in section 3 we give some algorithms for specific PAC learning problems. The results of section 3 also serve to distinguish our learning setting from other restrictions of the PAC setting in terms of what concept classes may be learned.) The following result makes the problem look more like one of designing a strategy for a cooperative game between the two learners, than a problem of approximating probability distributions. We show that if the outputs of hypotheses may be any real numbers, then the algorithm may be modified so that the hypotheses are probability distributions, and the separation is still PAC. There is no particular reason to suppose that probability distributions obtained in this way will constitute particularly good approximations to the underlying distributions generating the instances, according to previously-studied metrics, such as variation distance or KL-distance.

Theorem 1 *Let X be a domain of inputs. If there is a PAC algorithm in which each unsupervised learner may assign any real number to an element of X, then there is a PAC algorithm in which the learners must choose numbers that integrate or sum to 1 over the domain (i.e. are a probability distribution).*

Proof: Let $\mathcal{A}$ be an algorithm that returns a scoring function. So in a problem instance, $\mathcal{A}$ is applied twice, once to A's data and once to B's data, and we obtain functions $f_A : X \longrightarrow \mathbf{R}$ and $f_B : X \longrightarrow \mathbf{R}$. (So for example any $x \in X$ with $f_A(x) > f_B(x)$ would be labeled as positive by the overall hypothesis, under our convention that A receives the positive examples.)

Our approach is to re-scale any function returned by the algorithm so that the outcome of any comparison is preserved, but the new functions sum or integrate to 1. In a case where, for example, $\sum_{x \in X} f_A(x) = 1$ and $\sum_{x \in X} f_B(x) = 2$, this initially appears problematic: f_B has to be scaled down, but then the new values of f_B may become less than f_A. Note however that we can modify f_A by choosing an arbitrary $\hat{x}$ in A's data, and adding 1 to $f_A(\hat{x})$. This can only improve the resulting classifier (it may cause $\hat{x}$ to be put in A's class where previously it was put in B's class). Now the new f_A together with f_B can both be rescaled down by a factor of 2, and comparisons are clearly preserved.

Making the above idea general, suppose that algorithm $\mathcal{A}$ takes a sample of data S and returns a function $f : X \longrightarrow \mathbf{R}$. Modify $\mathcal{A}$ as follows. Define $g(x) = e^{f(x)}/(1 + e^{f(x)})$, so the range of g is $(0, 1)$. Let $P(X)$ be a probability distribution over X that does not vanish anywhere. Let $s = \sum_{x \in X} g(x)P(x)$, or $\int_{x \in X} g(x)P(x)dx$ for continuous X. s is well-defined and lies in the range $(0, 1)$. Now for a discrete domain X, the probability distribution returned by the modified $\mathcal{A}$ is $D'(x) = g(x)P(x)$ for all $x \in X$ except for some arbitrary $\hat{x} \in S$, where $D'(\hat{x}) = g(x)P(x) + 1 - s$. For a continuous domain the probability distribution is the mixture of the continuous density $D'(x) = g(x)P(x)$ with coefficient s and a point probability mass located at some $\hat{x} \in S$ with probability $1 - s$. □

As a consequence of the above result, we continue by giving algorithms for hypotheses that may output unrestricted real numbers.

Our other general result about the PAC learning framework we have defined provides some further information about how it compares with other variants of PAC learning in terms of which concept classes become learnable. First, note that the framework can be extended to a misclassification noise situation by letting each learner have examples that have been (correctly or incorrectly) assigned to the class label associated with that learner.

Theorem 2 *Any concept class that is PAC-learnable in the presence of uniform misclassification noise can be learned by two unsupervised learners in the presence of uniform misclassification noise if the input distribution is known to both learners.*

Proof: Let D be the known distribution over the input domain X. Let $\mathcal{C}$ be a concept class that is PAC-learnable with uniform misclassification noise. We may assume that $\mathcal{C}$ is closed under complementation (we have noted that if it is not closed under complementation we can take it closure under complementation which should still be PAC learnable).

Each learner takes a set S of N examples, where N is chosen such that a standard PAC algorithm would have error bound ϵ^2. Let D^+ (resp. D^-) be the probability that an example generated by D belongs to target T (resp. $X \setminus T$). Let ν be the noise rate. With

probability $\frac{(1-\nu)D^+}{(1-\nu)D^+ + \nu D^-}$ an example received by A belongs to target T; meanwhile with probability $\frac{(1-\nu)D^-}{(1-\nu)D^- + \nu D^+}$ an example received by B comes from $X \setminus T$.

Each learner labels all its examples as positive, and then generates a set of N examples from D, each of which is labeled positive with some probability $p < \frac{1}{2}$, otherwise negative. For learner A, the union of these two sets consists of a set of examples from a new probability distribution D', labeled by the same target concept T. It may be verified that the examples from T have misclassification noise with noise rate

$$\frac{(1-p)(D^+(1-\nu) + D^-\nu)}{(1-p)(D^+(1-\nu) + D^-\nu) + 1 - \nu}$$

and the examples from $X \setminus T$ have misclassification noise with noise rate

$$\frac{\nu + p(D^+(1-\nu) + D^-\nu)}{\nu + (D^+(1-\nu) + D^-\nu)}.$$

It may be verified from these expressions that there is a unique value of p in the range $[0, \frac{1}{2}]$ for which these two noise rates are equal, and for that value of p they are both strictly less than $\frac{1}{2}$.

Hence for some value of p we obtain data with uniform misclassification noise. An appropriate p can be found by trying all values $p = r\epsilon$ for $r = 0, \ldots, 1/2\epsilon$, and checking whether the hypothesis obtained (using a standard noise-tolerant PAC algorithm) is consistent with uniform misclassification noise.

The same reasoning applies to learner B using $X \setminus T$ as the target concept.

Let H be the set of examples labeled positive by the resulting hypothesis. Each learner assigns scores as follows. If the observed value of $D'(H)$ is at least $1 - \epsilon$, use a score of $\frac{1}{2}$ for all elements of X. Otherwise use a score of 1 for elements of H and a score of 0 for elements of $X \setminus H$.

Let D'_A and D'_B be the D''s for learners A and B.

$D(T)$ is the probability that a random example from D belongs to T. Assuming $D(T) < 1 - \epsilon$, we can say that error $O(\epsilon^2)$ with respect to D'_A implies error $O(\epsilon)$ with respect to D. If alternatively $D(T) \geq 1 - \epsilon/2$, the hypothesis H found by A will have a probability $D'(H) > 1 - \epsilon$ as observed on the data. A learner finding such a hypothesis then gives all examples a score of $\frac{1}{2}$, allowing B's scores to determine the overall classification. A similar argument applies for low values of $D(T)$ (where we expect B to assign scores of $\frac{1}{2}$). □

It is probably not the case that noise-free distribution-specific learnability with two unsupervised learners is actually equivalent to standard PAC-learnability with uniform misclassification noise. This is because, given the Noisy Parity Assumption (that it is hard to PAC-learn parity functions in the presence of random misclassification noise given the uniform distribution over input vectors), distribution-specific learning with two unsupervised learners is tractable (see section 3.3) in a situation where the uniform misclassification noise situation is intractable.

3 Examples of Concrete Learning Problems

The algorithms in this section give an idea of the new technical challenges, and also distinguish the learning setting from various others. So for example, the learnability result of section 3.3 distinguishes it from learnability with uniform misclassification noise or learnability with a restricted focus of attention, and the result of section 3.4 distinguishes it from learnability with one-sided error or learnability from positive or negative examples only.

3.1 Boolean Functions over a Constant Number of Variables

Let the domain X be vectors of length n of binary values 0/1 (where we use 0 to represent false and 1 to represent true). We show how to learn a function f from the class of boolean functions having the property that only k of the variables are relevant (affect the output value), where k is a constant. There are $< \binom{n}{k} 2^{2^k}$ functions in this class, so in the PAC framework, the brute-force approach of trying each one until a consistent hypothesis is found, is a polynomial-time algorithm.

We identify the following method for the two unsupervised learners framework. Each unsupervised learner constructs a scoring function $h : X \longrightarrow \{0, 1\}$ using the following rule. Let S be the sample of vectors that one of the learners has at its disposal. For any vector $\mathbf{v} \in X$, put $h(\mathbf{v}) = 1$ iff for all sets of k attributes in $\mathbf{v}$, the values they take in $\mathbf{v}$ equal the values they take in some element of S (and otherwise put $h(\mathbf{v}) = 0$).

The above rule can be expressed as a polynomial-sized boolean formula, and it can be seen moreover that the formula is satisfied by elements of S but not by any examples that were sent to the other unsupervised learner. (That follows from the fact that a boolean formula over k of the variables is being used to distinguish the two sets of boolean vectors.) Let ϕ_A and ϕ_B be the formulae constructed by learners A and B respectively. ϕ_A is PAC (being a consistent hypothesis of polynomial description length) and ϕ_B is PAC with respect to the complement of the target concept. Hence with high probability for $\mathbf{v}$ sampled using D, $\phi_A(\mathbf{v}) \neq \phi_B(\mathbf{v})$, and so a comparison of the values of $\phi_A(\mathbf{v})$ and $\phi_B(\mathbf{v})$ will usually (in the PAC sense) indicate correctly which learner to associate $\mathbf{v}$ with.

3.2 Monomials

Recall that a monomial is a boolean function consisting of the conjunction of a set of literals (where a literal is either a boolean attribute or its negation). Despite the simplicity of this class of functions, we have not resolved its learnability in the two unsupervised learners framework, even for monotone (*i.e.* negation-free) monomials. If the unsupervised learners are told which of them has the positive and which the negative examples, then the problem does have a simple solution (a property of any class of functions that is learnable from either positive examples only or else negative examples only). The "negative" unsupervised learner assigns a score of $\frac{1}{2}$ to all boolean vectors. The "positive" unsupervised learner uses its data to find a PAC hypothesis, and assigns a score of 1 to examples satisfying that hypothesis, and 0 to other examples.

3.2.1 Discussion of the Distribution-independent Learning Problem

Given a monomial f, let $pos(f)$ denote its satisfying assignments. The problem that arises when the unsupervised learners are not told which one is receiving the positive examples, is that the distribution over the negative examples could in fact produce boolean vectors that satisfy some monomial m that differs from target monomial t, but if $D(pos(m) \cap pos(t)) > \epsilon$ this may give excessive error. This problem can of course be handled in the special case where the monomial is over a constant number k of literals, and the algorithm of section 3.1 can be used. What happens then is that the unsupervised learners become more "conservative", tending to assign zero score more often to an instance, and in particular will do so for instances belonging to subsets of the domain where the subset corresponds to satisfying assignments of some k-variable monomial, and furthermore no instance has been observed in that subset. For monomials over only k literals, this approach does not excessively cut down the set of instances that are assigned a score of 1.

3.2.2 Learnability of Monomials over Vectors of Attributes Generated by a Product Distribution

In view of the importance of monomials as a concept class, we consider whether they are learnable given that the input distribution D belongs to a given class of probability distributions. This situation is intermediate between knowing D exactly (in which case by theorem 2 the problem would be solved since monomials are learnable in the presence of uniform misclassification noise) and the distribution-independent setting. We assume now that the class priors are known approximately, since we no longer have the equivalence of the one-button and two-button versions of PAC learnability. Formally, assume each learner can sample from D, but if the example drawn belongs to the other learner's class then the learner is told only that the example belonged to the other class, and no other information about it. Hence each learner has an "observed class prior", the observed probability that examples belong to one's own class.

Suppose that D is known to be a product distribution. Let $x_1, \ldots, x_n$ be the boolean attributes in examples. Let d_i, $i = 1, \ldots, n$, be the probability that the i-th attribute equals 1. For attribute x_i for which one of the literals x_i or $\overline{x_i}$ is in the target monomial t (we assume that they are not both in t), let p_i be the probability that the literal is satisfied, so $p_i = d_i$ for an un-negated literal, otherwise $p_i = 1 - d_i$.

We say that attribute x_i is "useful" if x_i or $\overline{x_i}$ is in t, and also $p_i \in [\epsilon, 1-\epsilon/n]$. Note that if $p_i < \epsilon$ then the probability that any example is positive is also $< \epsilon$, and if $p_i > 1 - \epsilon/n$ then only a very small fraction of examples can be negative due to their value of x_i.

The Algorithm

We use the fact that for D a product distribution, D restricted to the positive examples of a monomial is also a product distribution. We apply a test (step 3 of the algorithm) to see whether the observed data appears to come from a product distribution. The test identifies negative data when there is more than one useful attribute. Our scoring function separately handles the cases when < 2 attributes are useful.

1. Draw a sample S of size $O((n/\epsilon)^3 \log(\frac{1}{\delta}))$.

2. If the observed class prior of the examples is $\leq \epsilon/2$, reject all examples. Otherwise do the following.

3. (product distribution test) For each literal l that is satisfied by at least a fraction ϵ/n of elements of S, let S_l denote elements of S which satisfy l, and check whether the relative frequency with which any other literal is satisfied within S_l, differs from its frequency of being satisfied in S by at least ϵ/n^2. If so, the test "fails" and we assume that the negative examples are being seen, and give all examples a score of 1/2. Otherwise (the test is "passed"):

4. Let L be the set of literals satisfied by *all* elements of S.

   (a) If an example satisfies all the literals in L *and* fails to satisfy all literals that are satisfied by a fraction $< \epsilon/2n$ of elements of S, give that example a score of 1.

   (b) Otherwise, if the example still satisfies L, assign a score of 1/2.

   (c) Otherwise assign a score of 0.

Note: step 3 is a test with "one-sided error" in the sense that we may reasonably expect all product distributions to pass the test, but there exist distributions other than product distributions that may also pass. However, we show below that when a product distribution restricted to the negative data of a monomial passes the test, then there is at most one useful attribute.

3.2.3 Proving That the Algorithm is PAC

There must be at least one useful attribute in order for the frequency of both positive and negative examples to exceed ϵ. We consider two cases: first when there is only one useful attribute, second, when there is more than one.
Case 1: In this case, we expect the distributions over both the positive and negative examples to be close to product distributions, so that the test of step 3 of the algorithm will be passed in both A's case and B's case. Learner A gives a score of 1 to examples which satisfy the useful literal l, with the exception of a small fraction of them due to the additional requirement in step 4a. Meanwhile, learner B assigns a score of $\leq \frac{1}{2}$ to all examples satisfying l, since l is not satisfied by any of B's data. Hence learner A claims all but a fraction $< \epsilon/2$ of the positive data. By a similar argument, learner B claims all but a fraction $< \epsilon/2$ of the negative data.
Case 2: When there are two useful attributes, the positive examples are still generated by a product distribution, so A's data pass the test of step 3. Meanwhile, with probability $> 1 - \delta$, B's data fail this test, since when we choose literal l in target t that happens to be useful, and remove elements of S which satisfy l, then the conditional probability that any other useful literal is satisfied, changes by $> \epsilon/n^2$. (A Chernoff bound analysis assures that the change will be detected with probability $1 - \delta/2$.) All examples are then given scores of 1/2, and this allows A to claim positives and leave B the negatives.

3.3 Parity Functions

A parity function has an associated subset of the variables, and an associated "target parity" (even or odd), and evaluates to 1 provided that the parity of the number of "true" elements of that subset agrees with the target parity, otherwise the function evaluates to 0. This is the only concept class for which there is an "obvious" algorithm in this framework,

since parity functions have the distinction of being both learnable from positive examples only and from negative examples only. Similar to the algorithm of [11], each unsupervised learner finds the affine subspace of $GF(2)^n$ spanned by its examples, and assigns a score of 1 to elements of that subspace and a score of 0 to all elements of the domain.

Clearly these affine subspaces do not intersect, and if the two subspaces do not cover the whole domain, then the region not covered is with high probability a low-probability region.

3.4 Unions of Intervals

Let the domain be the real numbers $\mathbf{R}$, and assume that the target concept is a union of k intervals in $\mathbf{R}$. We show that this concept class is learnable by two unsupervised learners. This result shows that learnability with two unsupervised learners is distinct from learnability from positive examples only, or from negative examples only.

Each learner does the following. Let S be the set of real values that constitutes the data. Define scoring function f as

$$f(r) = -\Big(\min_{s \in S, s > r}(s) - \max_{s \in S, s < r}(s)\Big) \text{ if } r \notin S$$

$$f(r) = 1 \text{ if } r \in S.$$

This choice of scoring function ensures that when A's and B's scores are combined, the set of all points that are claimed by A consists of a union of at most k intervals, and this set contains A's data but not B's data. Hence we have a consistent hypothesis of V-C dimension no more than the target concept, so this method is PAC (with runtime polynomial in k, ϵ^{-1} and δ^{-1}). Note that the value of k needs to be prior knowledge, in contrast with PAC learning in the standard setting, where an appropriate sample size can be identified using the standard on-line approach of comparing the number of examples seen so far with the complexity of the simplest consistent classifier, and continuing until the ratio is large enough.

3.5 Rectangles in the Plane with Bounded Aspect Ratio

Let α denote the length of the target rectangle divided by the width, and our PAC-learning algorithm is polynomial in α as well as the standard PAC parameters. This simple concept class turns out to require a relatively complex method, and is worth studying on its own, to indicate the problems involved. Possible extensions are discussed in the next section. We do not have a PAC algorithm that works without the bound on the aspect ratio.

The general idea is that each learner partitions the domain into rectangles containing equal numbers of its data points, and given a query point $\mathbf{q}$, compares the coordinates of $\mathbf{q}$ with other points in the partition element within which $\mathbf{q}$ falls. A high score is given when there exist points in that partition element with similar coordinate values.
The Algorithm: Each learner does the following.

1. Generate a sample of size $N = O(\alpha \log(\delta^{-1})/\epsilon^9)$.

2. Build a partition P of the domain $\mathbf{R}^2$ as follows:

(a) Partition the domain into $1/\epsilon^2$ pieces using lines normal to the y-axis, such that each piece contains the same number of data points.[1]

(b) Partition each element of the above partition into $1/\epsilon^2$ pieces using lines normal to the x-axis, such that each piece contains the same number of data points.

3. For query point $\mathbf{q} \in \mathbf{R}^2$ the score assigned to $\mathbf{q}$ is computed as follows.

(a) Let $P_{\mathbf{q}} \in P$ be the rectangle in P containing $\mathbf{q}$.

(b) Let $S(P_{\mathbf{q}})$ be the sample points that lie inside $P_{\mathbf{q}}$.

(c) Sort $S(P_{\mathbf{q}}) \cup \{\mathbf{q}\}$ by x-coordinate. If $\mathbf{q}$ is among the first $(\frac{1}{\epsilon})^{-1}$ elements or among the last $(\frac{1}{\epsilon})^{-1}$ elements, then reject $\mathbf{q}$, *i.e.* assign $\mathbf{q}$ a score of $-\infty$ and terminate.

(d) If $\mathbf{q}$ was not rejected, define the x-cost of $\mathbf{q}$ to be the difference between the x-coordinates of the two neighbors of $\mathbf{q}$.

(e) Sort $S(P_{\mathbf{q}}) \cup \{\mathbf{q}\}$ by y-coordinate. If $\mathbf{q}$ is among the first $(\frac{1}{\epsilon})^{-1}$ elements or among the last $(\frac{1}{\epsilon})^{-1}$ elements, then reject $\mathbf{q}$.

(f) If $\mathbf{q}$ was not rejected, define the y-cost of $\mathbf{q}$ to be the difference between the y-coordinates of the two neighbors of $\mathbf{q}$.

(g) Finally, the score assigned to $\mathbf{q}$ is the negation of the sum of the x-cost and y-cost.

3.5.1 Proving That the Algorithm is PAC

Let P_A be the partition constructed by A and let P_B be the partition constructed by B. Let μ_A (respectively μ_B) be the measure on $\mathbf{R}^2$ induced by the input distribution D restricted to target T (respectively T', the complement of T) and re-normalised, so that we have $\mu_A(\mathbf{R}^2) = \mu_B(\mathbf{R}^2) = 1$. So, given region $R \subseteq \mathbf{R}^2$, $\mu_A(R)$ is the probability that a random input x lies within R conditioned on x being an element of T. Let $\hat{\mu}_A$ and $\hat{\mu}_B$ denote the measures μ_A and μ_B as observed by A and B respectively on the random examples used in the algorithm. The well-known V-C theory of [2] says that for a concept class $\mathcal{C}$ of V-C dimension v, given a sample of size[2] $O(v \log(\delta^{-1}\epsilon^{-1})/\epsilon)$, we have that with probability $1 - \delta$,

$$|\hat{\mu}(C) - \mu(C)| \leq \epsilon \text{ for all } C \in \mathcal{C}$$

where μ is a probability measure and $\hat{\mu}$ is the measure as observed on the sample. Noting that axis-aligned rectangles in $\mathbf{R}^2$ have V-C dimension 4, we deduce that if learners A and B draw samples of size $O(\log(\delta^{-1}\epsilon^{-1})/\epsilon)$, then with probability $1 - \delta$,

$$|\hat{\mu}_A(R) - \mu_A(R)| \leq \epsilon \text{ and}$$

$$|\hat{\mu}_B(R) - \mu_B(R)| \leq \epsilon, \text{ for all rectangles } R.$$

The following fact emerges automatically from the V-C bounds:

[1] Throughout we ignore rounding error in situations where for example an equal partition is impossible; such rounding will only change quantities by a constant.

[2] This is weaker than the known bound — we are using a weak bound to simplify the presentation.

Fact 3 *N is chosen such that if learners A and B use samples of size N, then with probability $1 - O(\delta)$ we have that for all rectangles R:*

$$|\hat{\mu}_A(R) - \mu_A(R)| \leq \epsilon^9$$

$$|\hat{\mu}_B(R) - \mu_B(R)| \leq \epsilon^9.$$

The following fact follows from the way the algorithm constructs the partition P:

Fact 4 *P_A and P_B are each of size $(1/\epsilon)^4$, and elements of P_A (respectively P_B) contain $(1/\epsilon)^5$ of A's (respectively B's) data points.*

From fact 4, given rectangle $R \in P_A$, $\hat{\mu}_A(R) = \epsilon^4$, and consequently $|\mu_A(R) - \epsilon^4| \leq \epsilon^9$ with high probability, using fact 3. Clearly all rectangles in P_A intersect target rectangle T (similarly members of P_B intersect T'). Now consider the potential problem of rectangles in P_B that contain positive examples. We continue by upper-bounding the number of those rectangles, and upper-bounding the amount of damage each one can do (due to claiming data examples that should be claimed by A).

Let $P_A^* \subseteq P_A$ be elements of P_A which intersect T' (so are not proper subsets of T). Similarly let P_B^* denote elements of P_B which intersect T. We show that the number of elements of P_A^* and P_B^* is substantially smaller than the cardinalities of P_A and P_B.

Fact 5 *Any axis-aligned line cuts $(\frac{1}{\epsilon})^2$ elements of P_A and similarly $(\frac{1}{\epsilon})^2$ elements of P_B.*

Corollary 6 *The boundary of T intersects at most $O((\frac{1}{\epsilon})^2)$ elements of P_A and similarly at most $O((\frac{1}{\epsilon})^2)$ elements of P_B.*

In particular, it intersects at most $4.(\frac{1}{\epsilon})^2$ elements of either partition.

So partition P_B has $(\frac{1}{\epsilon})^4$ elements each containing $(\frac{1}{\epsilon})^5$ data points, and only $O((\frac{1}{\epsilon})^2)$ of them intersect T. Now we consider how an element $R \in P_B$ could intersect T. We divide the kinds of overlap into

1. An edge overlap, where one edge and no vertices of T are overlapped by R.
2. A two-edge overlap, where 2 opposite edges and no corner of T are overlapped by R.
3. Any overlap where R contains a corner of T.

We treat these as separate cases. Note that since there are at most 4 overlaps of type 3 we may obtain relatively high bounds on the error they introduce, by comparison with the edge and two-edge overlaps, of which there may be up to $(\frac{1}{\epsilon})^2$ in the worst case. Throughout we use the following notation. Let $\mathbf{x}$ be a point in target rectangle T which is being assigned scores using A's and B's partitions. Let $\mathbf{x} \in$ rectangle $R_A \in P_A$ and $\mathbf{x} \in R_B \in P_B$, so that R_A intersects T.

Case 1: (edge overlap) R_B has an edge overlap with T. Consider steps 3c and 3e of the algorithm. When $\mathbf{x}$ is being compared with the points in R_B it will have either an x-coordinate or a y-coordinate which is maximal or minimal for data points observed in R_B. One of these steps of the algorithm will cause B to reject $\mathbf{x}$. But $\mathbf{x}$ will only have

a low probability (in particular $O(\epsilon^5)$) of having a maximal or minimal coordinate value amongst points in R_A (since R_A contains $(\frac{1}{\epsilon})^5$ data points and $\mathbf{x}$ is generated by the same distribution that generated those data points).

Case 2: (two-edge overlap) There are at most $(\frac{1}{\epsilon})^2$ two-edge overlaps possible. Suppose that in fact R_B overlaps the top and bottom edges of T (the following argument will apply also to the other sub-case). Hence all the two-edge overlaps do in fact overlap the top and bottom edges of T. Let x_T and y_T denote the lengths of T as measured in the x and y directions, so we have $x_T/y_T \in [1/\alpha, \alpha]$. Then the y-cost of R_B is at least y_T. Meanwhile, all but a fraction ϵ of boxes in P_A will give a y-cost of $\leq \epsilon \cdot y_T$. Also, all but a fraction ϵ of boxes in P_A will have x-costs at most $\epsilon \cdot x_T$. Using our aspect ratio assumption, this is at most $\epsilon\alpha y_T$. Hence, for points in all but a fraction ϵ of boxes in P_A, the y-cost will dominate, and the score assigned by B will exceed A's score.

Case 3: (corner overlap) Suppose R_B overlaps a corner of T. We show that R_B introduces error $O(\epsilon)$, and since there are at most 4 such rectangles, this case is then satisfactory. For R_B to introduce error $> \epsilon$, it must overlap a fraction $\Omega(\epsilon)$ of rectangles in P_A, hence $> (\frac{1}{\epsilon})^3$ rectangles in P_A. In this situation, R_B contains $\Omega((\frac{1}{\epsilon})^3)$ recangles in P_A in its interior. On average, both the x and y coordinates of sample points in these interior rectangles will be $\Omega((\frac{1}{\epsilon}))$ closer to each other than the points in R_B. This means that only an ϵ-fraction of points in these elements of P_A will have coordinates closer to points in R_B, than to some other point in the same element of P_A. Hence all but an ϵ-fraction of these points will be claimed by A.

3.5.2 Discussion, possible extensions

Obviously we would like to know whether it is possible to have PAC learnability without the restriction on the aspect ratio of the target rectangle. The restriction is arguably benign from a practical point of view. Alternatively, various reasonable "well-behavedness" restrictions on the input distribution would probably allow the removal of the aspect ratio restriction, and also allow simpler algorithms.

The extension of this result to unions of k rectangles in the plane is fairly straightforward, assuming that the aspect ratio restriction is that both the target region and its complement are expressible as a union of k rectangles all with bound α on the aspect ratio.

The general idea being used is likely to be extendable to any constant dimension, but then the case analysis (on the different ways that a partition element may intersect the region with the opposite class label) may need to be extended. If so it should generalize to unions of boxes[3] in fixed dimension (as studied in [4] in the setting of query learning, a generalization is studied in [3] in PAC learning). Finally, if boxes are PAC learnable with two unsupervised learners in time polynomial in the dimension, then this would imply learnability of monomials, considered previously.

[3] A *box* means the intersection of a set of halfspaces whose bounding hyperplanes are axis-aligned, *i.e.* each hyperplane is normal to one of the axes.

3.6 Linear Separators in the Plane

Given a set S of points in the plane, it would be valid for an unsupervised learner to use a probability distribution whose domain is the convex hull[4] of S, *provided that only a "small" fraction of elements of S are actually vertices of that convex hull.* For a general PAC algorithm we have to be able to handle the case when the convex hull has most or all of the points at its vertices, as can be expected to happen for an input distribution whose domain is the boundary of a circle, for example. Our general approach is to start out by computing the convex hull P and give maximal score to points inside P (which are guaranteed to have the same class label as the observed data). Then give an intermediate score to points in a polygon Q containing P, where Q has fewer edges. We argue that the way Q is chosen ensures that most points in Q are indeed claimed by the learner.

3.6.1 The Algorithm

The general idea is to choose a scoring function in such a way that we can show that the boundary between the classes is piecewise linear with $O(\sqrt{N})$ pieces, where N is sample size. This sublinear growth ensures a PAC guarantee, since we have an "Occam" hypothesis — the V-C dimension of piecewise linear separators in the plane with $O(\sqrt{N})$ pieces is itself $O(\sqrt{N})$.

1. Draw a sample S of size $N = O(\log(\delta^{-2}\epsilon^{-2})/\epsilon^2)$.
2. Let polygon P be the convex hull of S.
3. Let Q be a polygon having $\leq 2 + \sqrt{N}$ edges such that
   (a) Every edge of Q contains an edge of P
   (b) Adjacent edges of Q contain edges of P that are $\leq \sqrt{N}$ apart in the adjacency sequence of P's edges.
4. Define scoring function h as follows.
   (a) For points in P use a score of 1.
   (b) For each region contained between P and 2 adjacent edges of Q, give points in that region a score of the negation of the area of that region.
   (c) Reject all other points (not in Q).

Regarding step 3: Q can be found in polynomial time; we allow Q to have $2 + \sqrt{N}$ edges since P may have 2 acute vertices that force pairs of adjacent edges of Q to contain adjacent edges of P.

[4]The *convex hull* of a finite set S of points is the smallest convex polygon (more generally, polytope) that contains S. Clearly all the vertices of the convex hull of S are members of S.

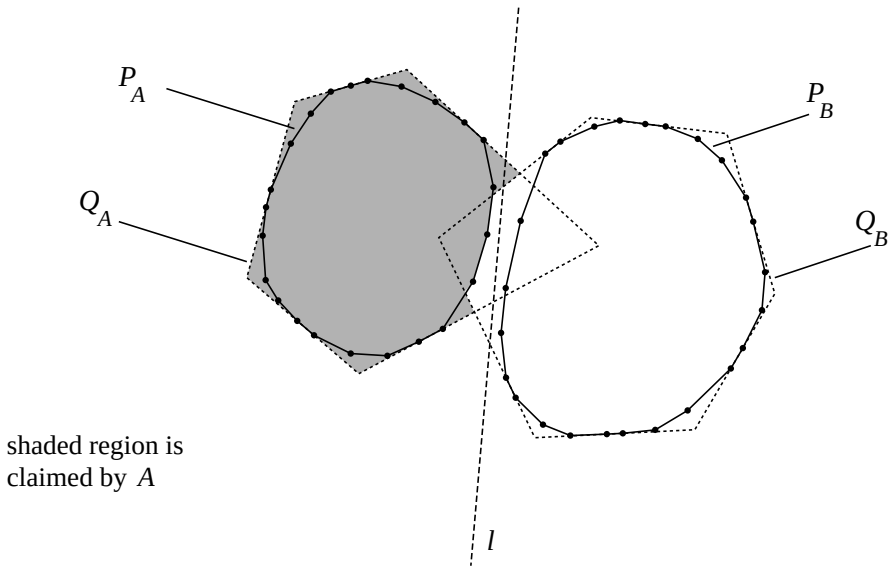

figure 1

3.6.2 Proving That the Algorithm is PAC

Figure 1 illustrates the construction. Let P_A and P_B be the convex hulls initially found by learners A and B respectively. They define subsets of the regions claimed by A and B respectively. Let Q_A and Q_B be the polygons Q constructed by A and B respectively. Let l be a line separating P_A and P_B. Observe that Q_A and Q_B respectively can each only have at most two edges that cross l, and at most one vertex on the opposite side of l from P_A and P_B respectively, using the fact that each edge of Q_A contains an edge of P_A, and similarly for Q_B and P_B.

Hence only one of the regions enclosed between P_A and two adjacent edges of Q_A can cross line l, and potentially be used to claim part of the interior of Q_B. If this region contains more than one of the similar regions in Q_B, then it will not in fact claim those regions of Q_B, since the score assigned to its interior will be lower. Omitting the details, it is not hard to show using these observations that the region claimed by A is enclosed by a polygon with $O(\sqrt{N})$ edges, and similarly for B. N was chosen such that the V-C bound of section 3.5.1 ensures PAC-ness with parameters ϵ and δ.

4 Conclusion and Open Problems

The standard requirement of PAC learning that algorithms must work for any input distribution D, appears to give rise to very novel algorithmic challenges, even for fairly elementary computational learning problems. At the same time however, the resulting algorithms do not appear to be applicable to the sort of class overlap situations that motivated the learning setting. Probably it will be necessary to model learning situations with an additional assumption that D should belong to some given class of distributions, as we did in

section 3.2.2. Our formalisation of this learning setting will hopefully provide insights into what assumptions need to be made about the distribution of inputs, in order for standard practical methods of unsupervised learning, such as kernel-based smoothing, to be applied.

The main open problem is the question of whether there is a concept class that is PAC-learnable but is not PAC-learnable in the two unsupervised learners framework. It would be remarkable if the two learning frameworks were equivalent, in view of the way the PAC criterion seems to impose a discipline of class separation on algorithms. Regarding specific concept classes, the most interesting one to get an answer for seems to be the class of monomials, which is a special case of nearly all boolean concept classes studied in the literature. We believe that it should be possible to extend the approach in section 3.2.2 to the assumption that D is a mixture of two product distributions, a class of distributions shown to be learnable in [5, 7].

Related open questions are: does there exist such a concept class for computationally unbounded learners (where the only issue is sufficiency of information contained in a polynomial-size sample). Also, can it be shown that *proper* PAC-learnability holds for some concept class but not in the two unsupervised learners version. (So, we have given algorithms for various learning problems that are known to be properly PAC-learnable, but the hypotheses we construct do not generally belong to the concept classes.)

With regard to the main open problem, we mention an obvious idea that fails. Take any concept class that is not PAC learnable given standard computational complexity theoretic assumptions, e.g. boolean circuits. Augment the representations of positive and negative examples with "random" numbers such that the number from any positive example and the number from any negative example between them allow an encoding of the target concept to be derived by a simple secret-sharing scheme, but the individual numbers are of no use. It looks as if this class should be PAC learnable, but should not be learnable in our framework. The problem is that we have to say whether examples labeled with the "wrong" numbers (corresponding to an alternative representation) are positive or negative. Then some probability distribution would be able to generate these examples instead of the ones we want to use.

References

[1] M. Anthony and N. Biggs (1992). *Computational Learning Theory*, Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, Cambridge.

[2] A. Blumer, A. Ehrenfeucht, D. Haussler and M.K. Warmuth (1989). Learnability and the Vapnik-Chervonenkis Dimension, J.ACM **36**, 929-965.

[3] N.H. Bshouty, S.A. Goldman, H.D. Mathias, S.Suri and H. Tamaki (1998). Noise-Tolerant Distribution-Free Learning of General Geometric Concepts. *Journal of the ACM* **45**(5), pp. 863-890.

[4] N.H. Bshouty, P.W. Goldberg, S.A. Goldman and H.D. Mathias (1999). Exact learning of discretized geometric concepts. *SIAM J. Comput.* **28**(2) pp. 674-699.

[5] M. Cryan, L. Goldberg and P. Goldberg (1998). Evolutionary Trees can be Learned in Polynomial Time in the Two-State General Markov Model. *Procs. of 39th FOCS symposium*, pp. 436-445.

[6] S. Dasgupta (1999). Learning mixtures of Gaussians. *40th IEEE Symposium on Foundations of Computer Science*.

[7] Y. Freund and Y. Mansour (1999). Estimating a mixture of two product distributions. *Procs. of 12th COLT conference*, pp. 53-62.

[8] A. Frieze, M. Jerrum and R. Kannan (1996). Learning Linear Transformations. *37th IEEE Symposium on Foundations of Computer Science*, pp. 359-368.

[9] V. Guruswami and A. Sahai (1999). Multiclass Learning, Boosting, and Error-Correcting Codes. *Procs. of 12th COLT conference*, pp. 145-155.

[10] D. Haussler, M. Kearns, N. Littlestone and M.K. Warmuth (1991). Equivalence of Models for Polynomial Learnability. *Information and Computation*, 95(2), pp. 129-161.

[11] D. Helmbold, R. Sloan and M.K. Warmuth (1992). Learning Integer Lattices. *SIAM Journal on Computing*, 21(2), pp. 240-266.

[12] M.J. Kearns (1993). Efficient Noise-Tolerant Learning From Statistical Queries, *Procs. of the 25th Annual Symposium on the Theory of Computing*, pp. 392-401.

[13] M. Kearns, Y. Mansour, D. Ron, R. Rubinfeld, R.E. Schapire and L. Sellie (1994). On the Learnability of Discrete Distributions, *Proceedings of the 26th Annual ACM Symposium on the Theory of Computing*, pp. 273-282.

[14] M.J. Kearns and R.E. Schapire (1994). Efficient Distribution-free Learning of Probabilistic Concepts, *Journal of Computer and System Sciences*, **48**(3) 464-497. (see also FOCS '90)

[15] J.C. Platt, N. Cristianini and J. Shawe-Taylor (2000). Large Margin DAGs for Multiclass Classification, *Procs. of 12th NIPS conference.*

[16] L.G. Valiant (1984). A Theory of the Learnable. *Commun. ACM* **27**(11), pp. 1134-1142.

[17] L.G. Valiant (1985). Learning disjunctions of conjunctions. *Procs. of 9th International Joint Conference on Artificial Intelligence.*