# Some Discriminant-based PAC Algorithms *

Paul W. Goldberg†
University of Warwick,
Department of Computer Science,
Coventry, CV4 7AL, UK.

January 5, 2006

**Abstract**

A classical approach in multi-class pattern classification is the following. Estimate probability distributions that generated the observations for each label class, and then label new instances by applying the Bayes classifier to the estimated distributions. That approach provides more useful information than just a class label; it also provides estimates of the conditional distribution of class labels, in situations where there is class overlap.

We would like to know whether it is harder to build accurate classifiers via this approach, than by techniques that may process all data with distinct labels together. In this paper we make that question precise by considering it in the context of PAC learnability. We propose two restrictions on the PAC learning framework that are intended to correspond with the above approach, and consider their relationship with standard PAC learning. Our main restriction of interest leads to some interesting algorithms that show that the restriction is not stronger (more restrictive) than various other well-known restrictions on PAC learning. An alternative slightly milder restriction turns out to be almost equivalent to unrestricted PAC learning.

**Keywords:** Computational learning theory, Pattern classification

*This work was supported by EPSRC Grant GR/R86188/01. This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the author's views. A preliminary version of this paper was presented at the 2001 COLT conference.

†author's email address: `pwg@dcs.warwick.ac.uk`

# 1 Introduction

We present some PAC learning algorithms for various learning problems, within a new restriction of the PAC setting. We begin by explaining the motivation for studying the new restriction, and continue with some general results about it, followed by the algorithms.

## 1.1 Background and Motivation

A standard approach to classification problems (see for example Duda and Hart (16), page 17) is the following. For each class, find a *discriminant function* that maps elements of the input domain to real values. These functions can be used to label any element $x$ of the domain with the class label whose associated discriminant function takes the largest value on $x$. The discriminant functions are usually estimates of the probability densities of points in each class, weighted by the class prior (relative frequency of that class), in which case we are using the *Bayes classifier*.

If it is possible to obtain good estimates of the probability distributions that generated the label classes, then (for reasons we explain below) these are often more useful than just an accurate classification rule. However, this raises the question of how much harder it becomes to learn to classify data well, if we actually insist on learning the distributions. This motivates our choice to study this general question in the context of PAC learning, since PAC learning gives a framework for results giving lower bounds on sample-size or computational requirements. These results allow different models of the learning process to be provably distinguished from each other, in terms of the learning problems that are tractable within each model.

Most recent work on pattern classification (for example work on support vectors (12)) of course does not try to learn label class distributions, but rather to find decision boundaries that optimize some performance guarantee, usually misclassification rate. Performance guarantees are derived from observed classification performance in conjunction with other features of the boundary such as syntactic or combinatorial complexity, or the number of support vectors and margin of separation. The general approach clearly requires examples with different labels to be taken in conjunction with each other when finding a decision boundary. By contrast, discriminant functions are constructed from individual label classes in isolation. It seems clearly "easier" to find a good classifier by considering all data together (so as to apply empirical risk minimisation), than by insisting that each label class must be independently converted into a discriminant function. Noting Vapnik's observation ((29), page 30) that one should not try to solve a problem via solving a more general problem, why exactly would we want to estimate the distributions of label classes?

The answer is that when the distributions can be found, the extra information that is obtained is often very useful in practice. In contrast with decision boundaries, we obtain for a domain element $x$, the values of the probability densities of label classes at $x$, which provide a conditional distribution over the class label of $x$. A predicted class label for $x$ can then take into account variable misclassification penalties, or changes in the assumed class priors. There are of course other ways to obtain such distributions, for example using logistic regression, or more generally (for $k$-class classification) neural networks with $k$ real-valued outputs re-scaled using the softmax activation function (see Bishop (6) for details). However, unsupervised learning for each class – if it can be done successfully – has other advantages over these techniques, such as the following.

1. Label classes can be added without re-training the system. So for example if a new symbol were added to a character set, then given a good estimate of the probability distribution over images of the new symbol, this can be used in conjunction with pre-existing models for how the other symbols are generated.

2. *Outlying* instances are those that lie in regions of the domain where the distributions have low weight. We usually cannot assign a sensible label to such instances, however they may at least be recognised as a result of all class label distributions having very small weight around such an instance.

3. For applications such as handwritten digit recognition, it is arguably more natural — from the perspective of cognitive modeling — to model the data generation process in terms of 10 separate probability distributions, than as a collection of thresholds between different digits. This is because a handwritten zero (say) is nearly always the result of a process that first chooses the label "0" and then creates the image. It is not the result of a process that first generates a character and then assigns it the label "0" based on context, appearance or other criteria.

Another difficulty with decision boundaries arises specifically in the context of multiclass classification. It has been noted (1) that multiclass classifiers are often constructed using multiple 2-class classifiers. How to combine them is a challenging topic that has itself received much recent attention, see for example (19; 1). In practical studies such as (27) 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 is 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 1-v-1 classification (as studied in (27)) raises the problem of combining the collection of pairwise classifiers in a principled way to get an overall classification, for example ensuring that all classes are treated the same way. In (27), 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.)

With regard to PAC learning, the approach of applying unsupervised learning to each label class, can treat situations where class overlap occurs (as is usually the case in practice). Standard PAC algorithms do not address this problem (although there have been extensions such as "probabilistic concepts" (24) that do so, and methods using support vectors that also allow decision boundaries that do not necessarily agree with all observed data). It is not hard to verify (see (26)) that when we have good estimates of the class label distributions (in a sense described below in Section 1.3) then the associated classifier is approximately optimal in the agnostic PAC sense. For large data set sizes, it becomes feasible to find good estimates of these distributions, and obtain this more useful "summary" of the data.

The algorithms described in this paper are given in the context of simple 2-class classification, as opposed to multi-class classification. This is because we aim to explore the problems arising from an insistence upon treating each label class independently. The algorithms would however apply in a multi-class context where each pair of classes is separated by a boundary belonging to the given set of boundaries.

## 1.2 Formal Definition of the Learning Framework

In PAC learning (see for example (3) for a detailed introduction) there is a source of data consisting of instances generated by a probability distribution $D$ over a domain $X$, labeled using an unknown "target function" $t : X \longrightarrow \{0,1\}$. The objective is to find a classifier $h : X \longrightarrow \{0,1\}$ which is a good approximation to $t$ with respect to probability measure $D$. As usual we will let $\epsilon$ denote a misclassification rate (probability that $t$ and $h$ disagree on random $x$) and $\delta$ denote the uncertainty (probability that error rate $\epsilon$ is not attained). We refer to the members of $t^{-1}(1)$ as the "positive examples" and the members of $t^{-1}(0)$ as the "negative examples".

We say that a set $\mathcal{C}$ of functions from $X$ to $\{0,1\}$ (the "concept class") is PAC learnable if there exists an algorithm $\mathcal{A}$ that for any $t \in \mathcal{C}$, with probability at least $1-\delta$ outputs $h : X \longrightarrow \{0,1\}$ having misclassification rate at most $\epsilon$. $\mathcal{A}$ is required to run in time polynomial in $\epsilon^{-1}$ and $\delta^{-1}$ and other parameters (usually, the syntactic description length of $t$ and members of the training data). $\mathcal{A}$ may sample $x$ from $D$ in unit time, and obtain $(x, t(x))$. In this standard definition, we assume that $t$ and $D$ may be worst-case (chosen by an adversary).

**Notation.** For $\ell \in \{0,1\}$ let $D_\ell$ denote the restriction of $D$ to $t^{-1}(\ell)$. Let $p_\ell$ denote the class prior for label $\ell$, $p_\ell = \Pr_{x \sim D}(t(x) = \ell)$. We assume throughout that $p_\ell > 0$. Thus

$$\begin{array}{ll} \Pr_{x \sim D_\ell}(x) = \frac{1}{p_\ell} \Pr_{x \sim D}(x) & \text{for } t(x) = \ell \\ \Pr_{x \sim D_\ell}(x) = 0 & \text{for } t(x) = 1 - \ell \end{array}$$

For any probability distribution $P$ over $X$ (for example $D$ or $D_\ell$), an algorithm *with access to* $P$ may in unit time draw an unlabeled sample from $P$.

It is shown in (20) that the standard PAC framework is equivalent to a "two-button" version, where an algorithm has access to a "positive example oracle" and a "negative example oracle". (The two-button version conceals the class priors and only gives the algorithm access to the distribution as restricted to each class label. Thus the oracles generate examples from $D_1$ and $D_0$ respectively.) We define a restriction of "two-button" learning as follows.

**Definition 1** *Suppose algorithm $\mathcal{A}$ has access to a distribution $P$ over $X$, and the output of $\mathcal{A}$ is a function $f : X \longrightarrow \mathbb{R}$. Execute $\mathcal{A}$ twice, using $D_1$ (respectively $D_0$) for $P$. Let $f_1$ and $f_0$ be the functions obtained respectively. For $x \in X$ let*

$$\begin{array}{rll} h(x) = 1 & \text{if} & f_1(x) > f_0(x) \\ h(x) = 0 & \text{if} & f_1(x) < f_0(x) \\ h(x) \text{ undefined} & \text{if} & f_1(x) = f_0(x) \end{array}$$

*If $\mathcal{A}$ takes time polynomial in $\epsilon^{-1}$ and $\delta^{-1}$, and $h$ is PAC with respect to $\epsilon$ and $\delta$, then we will say that $\mathcal{A}$ PAC-learns via discriminant functions.*

It is clear that if $\mathcal{A}$ can be found such that the resulting $h$ is PAC, then we have PAC learnability in the two-button setting, and hence standard PAC learnability.

In specifying the restriction above, we are keeping it both simple and "severe", in the sense of making it difficult to find algorithms within the restricted framework. This is with a view to getting strong positive results, and also to maximizing the potential for negative results (PAC learnable problems that are hard within the restricted setting). One could devise less severe restrictions to capture the general idea of learning via discriminant functions. Alternatives are discussed at the end of this section.

We also consider the following slightly less severe variant related to POSEX learnability as introduced in Denis (15), in which $\mathcal{A}$ has access to $D$ (in (15) the "EX" oracle), in addition to $D_1$ (in (15) the "POS" oracle). This is formalized as Definition 2, and it turns out that we can be much more specific about learnability in this sense, namely it is intermediate between PAC learnability with uniform misclassification noise and basic PAC learnability.

**Definition 2** *Suppose algorithm $\mathcal{A}$ has access to $D$ in addition to distribution $P$ over $X$, and the output of $\mathcal{A}$ is a function $f : X \longrightarrow \mathbb{R}$. Execute $\mathcal{A}$ twice, using $D_1$ (respectively $D_0$) for $P$. Let $f_1$*

*and $f_0$ be the functions obtained respectively. For $x \in X$ let*

$$\begin{array}{rll} h(x) = 1 & \text{if} & f_1(x) > f_0(x) \\ h(x) = 0 & \text{if} & f_1(x) < f_0(x) \\ h(x) \text{ undefined} & \text{if} & f_1(x) = f_0(x) \end{array}$$

*If $\mathcal{A}$ takes time polynomial in $\epsilon^{-1}$ and $\delta^{-1}$, and $h$ is PAC with respect to $\epsilon$ and $\delta$, then we will say that $\mathcal{A}$ PAC-learns via discriminant functions with access to $D$.*

POSEX learnability requires that $f_1$ be a 0/1-valued function that constitutes a PAC hypothesis. We discuss the relationship in more detail in the next section.

**Notation.** We refer to the two instances of the unsupervised learning algorithm as $\mathcal{A}_1$ and $\mathcal{A}_0$, so that $\mathcal{A}_1$ consists of $\mathcal{A}$ with the (unlabeled) positive data as input, and $\mathcal{A}_0$ is $\mathcal{A}$ with the (unlabeled) negative data as input. (Note however that learning according to Definition 1 or 2 does not allow $\mathcal{A}_\ell$ to use the value of $\ell$.) We let $S_\ell$ denote the sample of unlabeled data drawn by $\mathcal{A}_\ell$.

Let us conclude this section by considering alternative restrictions to PAC learnability that capture the informal notion of learning via discriminant functions. We could for example ask that $f_1$ and $f_0$ should be probability distributions. Our reason for not doing so is that, while we believe that the functions produced by our algorithms can be re-scaled on an ad-hoc basis to become probability distributions, there is no guarantee in the distribution-free PAC setting that they are similar to the unknown distributions $D_1$ and $D_0$.

Another natural candidate is a variant in which $\mathcal{A}_1$ and $\mathcal{A}_0$ are two distinct agents that know their class labels. Equivalently, we would be seeking two algorithms $\mathcal{A}_1$ and $\mathcal{A}_0$ that respectively have access to the positive and negative data (equivalently, an algorithm $\mathcal{A}_\ell$ that may use the value $\ell$). Observe however that this relaxation is not helpful for any concept class $\mathcal{C}$ that is closed under complement (meaning that for all $c \in \mathcal{C}$, we also have $X \setminus c \in \mathcal{C}$). Consequently, any algorithms that require this relaxation would need to exploit additional prior information about data from different classes. There's nothing wrong with that of course, but it departs from our focus on the approach of independently processing each label class.

### 1.3 Related Work

There has been much work on the comparison of alternative notions of PAC learning with each other, with the criterion for distinguishability being that some learning task (concept class) should be tractable[1] in one variant but not in the other. The early paper of Haussler et al. (20) showed that various alternative definitions of PAC learnability are equivalent in this sense. Examples of distinctions that have been found between different restrictions to the framework include learning with a restricted focus of attention (4; 5) which is shown to be more severe that learnability in the presence of uniform misclassification noise. Learnability with Statistical Queries (22) is also known to be at least as severe as learnability with uniform misclassification noise. Perhaps the most important result of this kind is the equivalence between PAC learnability and "weak" PAC learnability found by Schapire (28) which led to the development of boosting techniques. Blum (7) exhibits a concept class that distinguishes PAC learnability from mistake-bound learning, and that is of interest here since we use the same concept class (in Section 3.3) to show that our restriction of PAC learnability is likewise distinct from mistake-bound learnability.

We noted that learning under the constraint of Definition 2 is related to POSEX learning. Specifically we have:

[1] "tractable" is usually taken to mean that the computational and sample-size requirements for learning, should be polynomial in the parameters of the task.

**Observation 1** *If a concept class and its complement are POSEX learnable, then they are learnable under Definition 2.*

***Proof.*** (sketch; the following technique is used in more detail in Theorem 1) Let $\mathcal{A}_P$ be a POSEX algorithm for $\mathcal{C}$ and let $\mathcal{A}'_P$ be a POSEX algorithm for $\mathcal{C}'$, the class of sets whose complements are members of $\mathcal{C}$. An algorithm $\mathcal{A}$ in the sense of Definition 2 works as follows. $\mathcal{A}$ can sample from $D$, and it uses samples from $D_\ell$ as "positive" examples. $\mathcal{A}$ applies $\mathcal{A}_P$ and $\mathcal{A}'_P$ to this data, and at least one of the two hypotheses $h, h'$ is PAC, since the "positive" data really is positive from the perspective of one of $\mathcal{A}_P$ and $\mathcal{A}'_P$.

$h$ and $h'$ are then tested on further data; the chosen hypothesis should contain almost all samples from $D_\ell$, and if both $h$ and $h'$ do so, choose the one that contains fewer samples from $D$. That hypothesis with high probability maps samples from $D_\ell$ to 1 and samples from $D_{1-\ell}$ to 0. Let $\mathcal{A}_1$ and $\mathcal{A}_0$ be the two runs of $\mathcal{A}$ having access to $D_1$ and $D_0$ respectively. The overall hypothesis is obtained from two functions found by $\mathcal{A}_1$ and $\mathcal{A}_0$, and its misclassification rate is at most the sum of their error rates. □

In Section 2 we show that if a concept class is learnable in the presence of noise, then it and its complement are POSEX learnable, and hence by the above observation, learnable under Definition 2. The result of Section 2 answers a question raised by Letouzey et al. (25) (the hierarchies of inclusions given in Equations (4,5) of (25) can be merged).

There are some interesting algorithms having PAC-like performance guarantees for learning probability distributions; the topic was introduced in (23), see also (13; 17; 18; 14). The criterion for learning a distribution $D$ is to obtain a hypothesis distribution which is within $\epsilon$ of $D$ under some measure of similarity. The KL distance and the variation distance are usually considered. We noted above that when distributions are learned under these criteria, the Bayes classifier achieves agnostic PAC-ness. However, the algorithms we describe here differ substantially from these previous ones (as well as from the algorithms in the much more extensive general literature on unsupervised learning). The reason is that our aim is not really to approximate a distribution over inputs. Rather, it is to construct a discriminant function in such a way that we expect it to work well in conjunction with the corresponding discriminant function constructed on data with the opposite class label.

### 1.4 Summary of Results

The rest of the paper is organized as follows. In Section 2 we consider learning under Definition 2 in which $\mathcal{A}_\ell$ has access to $D$ in addition to $D_\ell$. We show that PAC learnability with uniform misclassification noise implies PAC learnability with discriminant functions and access to $D$. This gives us a good understanding of how learning under Definition 2 fits in with other restrictions.

In Section 3 we move to the trickier issue of learning according to Definition 1. We exhibit algorithms that show that the restriction is not more severe than various other restrictions of PAC learning. In Section 3.1 we show that parity functions are learnable in this setting, which distinguishes it from learnability in the presence of noise (subject to the "noisy parity assumption", which is the widespread assumption that parity functions are hard to learn in the presence of uniform misclassification noise) as well as Statistical Query (SQ) learnability (22) (since it is known from (22) that parity functions are not learnable using SQs.) In Section 3.2 we distinguish the setting from learnability from positive data only (or negative data only) by studying the class of unions of intervals on the real line. In Section 3.3 we distinguish the setting from mistake-bound learning, using a concept class from (7).

The two remaining algorithms indicate some limits to our success at finding algorithms in the restricted setting. Section 3.4 shows how to learn linear separators in the plane, using an approach that we have not been able to extend to three or more dimensions. In Section 3.5 we show how to learn monomials provided that the input distribution $D$ is an unknown product distribution. Learning under this sort of assumption is in a sense intermediate between learning with access to $D$ (Definition 2) and learning via discriminant functions in the sense of Definition 1.

## 2 Learning via Discriminant Functions with Access to $D$

In this section we show that given a standard noise-tolerant PAC learning algorithm, we may use it to construct an algorithm for POSEX learning and hence learning in the restriction of Definition 2. We do this in two stages — Proposition 1 shows how this is achieved provided that an estimate of the class prior $p_\ell$ is provided to $\mathcal{A}_\ell$, and Theorem 1 extends the result to the setting in which the class priors are unknown.

Here is an overview of the proofs in this section. Proposition 1 analyses learning with a given noise rate. This uses a standard definition of noise-tolerant PAC learning in which an algorithm $\mathcal{A}^\eta$ has parameter $\eta$. $\eta$ is the probability that an example is mislabeled; $0 \le \eta < \frac{1}{2}$. $\mathcal{A}^\eta$ should then be polynomial in $(\frac{1}{2} - \eta)^{-1}$ in addition to other parameters. Given the class prior $p_\ell$ (the probability that a random instance has label $\ell$) we can generate random samples from a fixed distribution that is a mixture of $D$ and $D_\ell$, such that they have uniform misclassification rate, which allows $\mathcal{A}^\eta$ to be used. In fact, we show that an additive approximation $\overline{p_\ell}$ can be used in place of $p_\ell$. This is done by exhibiting a coupling of the two labeled sample distributions (one using $p_\ell$ and the other using $\overline{p_\ell}$), in which they are very likely to generate the same data.

In Theorem 1 we exploit the fact that an approximation $\overline{p_\ell}$ can be used. The algorithm of Figure 2 tries out a sequence of values for $\overline{p_\ell}$, at least one of which is a good approximation to $p_\ell$. The previous algorithm is used for each of these values, which generates a collection of hypotheses, and the empirically best hypothesis is shown to be PAC.

**Proposition 1** *Let $\mathcal{A}^\eta$ be an algorithm with parameter $\eta$, $0 \le \eta < \frac{1}{2}$, that has access to labeled data, where elements of $X$ are distributed according to $D$, with a uniform label noise rate of $\eta$. Suppose that $\mathcal{A}^\eta$ uses time $p(\epsilon^{-1}, \delta^{-1}, (\frac{1}{2} - \eta)^{-1})$ (where $p$ is some polynomial), and with probability at least $1 - \delta$ returns a hypothesis having error at most $\epsilon$ (with respect to $D$).*

*For $\ell \in \{0,1\}$ let $p_\ell = \Pr_{x\sim D}(t(x) = \ell)$. Suppose that $|\overline{p_\ell} - p_\ell| \le \Delta/p(\epsilon^{-1}, \delta^{-1}, (\frac{1}{2} - \eta)^{-1})$. ($\Delta \in [0,1]$.) Suppose that the algorithm of Figure 1 is executed with input $\overline{p_\ell}$ and access to $D_\ell$ and $D$. Then with probability $1 - \delta - \Delta$, the algorithm outputs $f_\ell : X \longrightarrow \{0,1\}$ satisfying*

$$\Pr_{x\sim\frac{1}{2}(D+D_\ell)}(f_\ell(x) \ne t(x)) \le \epsilon \quad \text{for} \quad \ell = 1$$
$$\Pr_{x\sim\frac{1}{2}(D+D_\ell)}(f_\ell(x) \ne 1 - t(x)) \le \epsilon \quad \text{for} \quad \ell = 0$$

**Comment.** The fact that $f_\ell$ has error at most $\epsilon$ for $x \sim \frac{1}{2}(D + D_\ell)$ implies that $f_\ell$ has error at most $2\epsilon$ for $x \sim D$.

***Proof.*** We may assume that the concept class $\mathcal{C}$ is closed under complementation, since if $\mathcal{C}$ is learnable with misclassification noise then its closure under complementation is also learnable under misclassification noise.

Since $\mathcal{C}$ is closed under complementation, it suffices by symmetry to show that $f_1$ satisfies: with probability at least $1 - \delta - \Delta$, $\Pr_{x\sim\frac{1}{2}(D+D_1)}(f_1(x) \ne t(x)) \le \epsilon$.

*Input $\overline{p_\ell}$, an estimate of $\mathcal{A}_\ell$'s class prior $p_\ell$ ($\ell \in \{0,1\}$).*
*Let $\eta = \frac{\overline{p_\ell}}{2\overline{p_\ell}+1}$; $N = p(\epsilon^{-1}, \delta^{-1}, (\frac{1}{2} - \eta)^{-1})$.*

1. *Construct a labeled sample $S_\ell$ as follows. For $m = 1\ldots, N$ do:*
   (a) *Let $c_m$ be a "fair coin" random variable; $c_m = 0$ or $1$ with probability $\frac{1}{2}$; let $\ell_m$ be a 0/1 random variable, $\ell_m = 1$ with probability $\frac{\overline{p_\ell}}{2\overline{p_\ell}+1}$.*
   (b) *If $c_m = 1$, sample $x$ from $D$ and let $(x, \ell_m) \in S_\ell$.*
   (c) *If $c_m = 0$, sample $x$ from $D_\ell$ and let $(x, 1) \in S_\ell$.*
2. *Input $S_\ell$ to $\mathcal{A}^\eta$ using $\eta = \frac{\overline{p_\ell}}{2\overline{p_\ell}+1}$ to obtain a hypothesis $h_\ell : X \longrightarrow \{0,1\}$.*
3. *$f_\ell(x) = h_\ell(x)$ for all $x \in X$.*

Figure 1: Learning in the sense of Definition 2 using noise-tolerant PAC algorithm and estimates of class priors

Let $(x, j)$ be the element of $S_1$ constructed on the $m$-th iteration.

$$\begin{array}{rcl} \Pr(t(x) = 0) & = & \Pr(c_m = 1)\Pr_{x\sim D}(t(x) = 0) = \frac{1}{2}(1 - p_1) \\ \Pr(t(x) = 1) & = & 1 - \frac{1}{2}(1 - p_1) = \frac{1}{2}(1 + p_1) \end{array} \tag{1}$$

Next we give expressions for misclassification rates $\Pr(j = 0 \mid t(x) = 1)$ and $\Pr(j = 1 \mid t(x) = 0)$. Consider first the case that $t(x) = 1$. Note that

$$\begin{array}{rl} \Pr(j = 0 \mid t(x) = 1) = & \Pr(j = 0 \mid t(x) = 1 \wedge c_m = 0)\Pr(c_m = 0 \mid t(x) = 1) \\ & + \Pr(j = 0 \mid t(x) = 1 \wedge c_m = 1)\Pr(c_m = 1 \mid t(x) = 1). \end{array}$$

$\Pr(j = 0 \mid t(x) = 1 \wedge c_m = 0) = 0$, since if $c_m = 0$ then Step (1c) assigns label 1. Hence

$$\Pr(j = 0 \mid t(x) = 1) = \Pr(j = 0 \mid t(x) = 1 \wedge c_m = 1)\Pr(c_m = 1 \mid t(x) = 1). \tag{2}$$

When $c_m = 1$, we have $j = \ell_m$ where $\ell_m = 1$ with probability $\overline{p_1}/(2\overline{p_1} + 1)$, so

$$\Pr(j = 0 \mid t(x) = 1 \wedge c_m = 1) = 1 - \frac{\overline{p_1}}{2\overline{p_1} + 1} = \frac{\overline{p_1} + 1}{2\overline{p_1} + 1}. \tag{3}$$

$$\Pr(c_m = 1 \mid t(x) = 1) = \frac{\Pr(c_m = 1)\Pr(t(x) = 1 \mid c_m = 1)}{\Pr(t(x) = 1)} = \frac{\frac{1}{2}p_1}{\Pr(t(x) = 1)}.$$

$\Pr(t(x) = 1) = \frac{1}{2}(1 + p_1)$ by Equation (1), hence

$$\Pr(c_m = 1 \mid t(x) = 1) = \frac{\frac{1}{2}p_1}{\frac{1}{2}(1 + p_1)} = \frac{p_1}{1 + p_1}. \tag{4}$$

Hence from Equations (2) and (3) and (4),

$$\Pr(j = 0 \mid t(x) = 1) = \Big(\frac{\overline{p_1} + 1}{2\overline{p_1} + 1}\Big)\Big(\frac{p_1}{1 + p_1}\Big).$$

Now consider the case that $t(x) = 0$ (where $(x, j)$ is the labeled exampled constructed on the $m$-th iteration of $\mathcal{A}_1$); note that

$$\begin{aligned}\Pr(j = 1 \mid t(x) = 0) = \ & \Pr(j = 1 \mid t(x) = 0 \wedge c_m = 0) \Pr(c_m = 0 \mid t(x) = 0) \\ & + \Pr(j = 1 \mid t(x) = 0 \wedge c_m = 1) \Pr(c_m = 1 \mid t(x) = 0).\end{aligned}$$

If $c_m = 0$ then from Step (1c), $t(x) = 1$. Hence

$$\begin{aligned}\Pr(c_m = 1 \mid t(x) = 0) &= 1 \\ \Pr(c_m = 0 \mid t(x) = 0) &= 0.\end{aligned}$$

Consequently,

$$\Pr(j = 1 \mid t(x) = 0) = \Pr(j = 1 \mid t(x) = 0 \wedge c_m = 1). \tag{5}$$

When $c_m = 1$ we have $j = \ell_m$ where $\ell_m = 1$ with probability $\overline{p_1}/(2\overline{p_1} + 1)$, so

$$\Pr(j = 1 \mid t(x) = 0 \wedge c_m = 1) = \frac{\overline{p_1}}{2\overline{p_1} + 1}. \tag{6}$$

From (5) and (6),

$$\Pr(j = 1 \mid t(x) = 0) = \frac{\overline{p_1}}{2\overline{p_1} + 1}.$$

Based on the above expressions for the misclassification rates $\Pr(j = 0 \mid t(x) = 1)$ and $\Pr(j = 1 \mid t(x) = 0)$, and noting that $N$ is defined in Figure 1, Step (1) of the algorithm of Figure 1 is equivalent to the following:

1. For $m = 1 \dots N$ do:
   (a) Sample $x$ from the mixture $\frac{1}{2}(D + D_1)$.
   (b) Sample $r_m$ uniformly at random from $[0, 1]$.
   (c) If $t(x) = 1$, then if $r_m < (\frac{\overline{p_1}+1}{2\overline{p_1}+1})(\frac{p_1}{1+p_1})$ label $x$ incorrectly else label $x$ correctly.
   (d) If $t(x) = 0$, then if $r_m < \frac{\overline{p_1}}{2\overline{p_1}+1}$ label $x$ incorrectly else label $x$ correctly.

Let $D_{\mathcal{A}}$ be the above distribution over samples of size $N$. Let $D_\eta$ be the following distribution over samples of size $N$:

1. For $m = 1 \dots N$ do:
   (a) Sample $x$ from the mixture $\frac{1}{2}(D + D_1)$.
   (b) Sample $r_m$ uniformly at random from $[0, 1]$.
   (c) If $r_m < \frac{\overline{p_1}}{2\overline{p_1}+1}$ label $x$ incorrectly else label $x$ correctly.

Let $S_N$ denote the set of labeled samples of size $N$. Define a distribution $D_{2N}$ over $S_N \times S_N$ by using the same sequence of $x$'s and $r_m$ and the above procedures for constructing the labeled samples, so that the two marginal distributions over $S_N$ are $D_{\mathcal{A}}$ and $D_\eta$. For $(S, S') \sim D_{2N}$,

$$\Pr(S \neq S') \leq N \cdot \Pr_{x \sim \frac{1}{2}(D+D_1)}(t(x) = 1) \cdot \left| \left(\frac{\overline{p_1} + 1}{2\overline{p_1} + 1}\right)\left(\frac{p_1}{1 + p_1}\right) - \frac{\overline{p_1}}{2\overline{p_1} + 1} \right|.$$

This is because there are $N$ opportunities for $S$ to differ from $S'$, and this occurs when $x$ sampled from $\frac{1}{2}(D+D')$ satisfies $t(x)=1$. In that case, the labels will differ when $r_m$ lies between $\frac{\overline{p_1}}{2\overline{p_1}+1}$ and $(\frac{\overline{p_1}+1}{2\overline{p_1}+1})(\frac{p_1}{1+p_1})$. Consequently,

$$\Pr(S \neq S') \leq N \cdot \frac{|\overline{p_1} - p_1(\frac{1+\overline{p_1}}{1+p_1})|}{2\overline{p_1}+1} \leq N \cdot (2\overline{p_1}+1)^{-1} \cdot |\overline{p_1} - p_1| \leq N|\overline{p_1} - p_1|.$$

By definition of $\mathcal{A}^\eta$, for $S \sim D_\eta$, $N = p(\epsilon^{-1}, \delta^{-1}, (\frac{1}{2}-\eta)^{-1})$, $\eta = \frac{\overline{p_1}}{2\overline{p_1}+1}$, with probability $1-\delta$, $\mathcal{A}^\eta$ on input $S$ returns $h'$ having error $\Pr(h'(x) \neq t(x)) \leq \epsilon$ for $x \sim \frac{1}{2}(D+D_1)$.

Hence for $S \sim D_{\mathcal{A}}$, $N = p(\epsilon^{-1}, \delta^{-1}, (\frac{1}{2}-\eta)^{-1})$, with probability $1-\delta-N(|p_1-\overline{p_1}|)$, $\mathcal{A}^\eta$ on input $S$ returns $h'$ having error $\Pr(h'(x) \neq t(x)) \leq \epsilon$ for $x \sim \frac{1}{2}(D+D_1)$. Given our assumption that $|p_1 - \overline{p_1}| \leq \Delta/N = \Delta/p(\epsilon^{-1}, \delta^{-1}, (\frac{1}{2}-\eta)^{-1})$, the result follows. $\square$

1. *Let* $p(\epsilon,\delta) = \max_{0\leq\eta\leq 1/3} p(\epsilon^{-1}, \delta^{-1}, (\frac{1}{2}-\eta)^{-1})$*; where* $p(\cdot,\cdot,\cdot)$ *is the sample size in terms of error, uncertainty and noise rate used by* $\mathcal{A}^\eta$ *in Figure 1; Let* $\Delta = \delta/32p(\epsilon,\delta)$*;* $N = (16/\epsilon)^2 \log(128p(\epsilon,\delta)/\delta^2)$*;* $\mathcal{H} = \emptyset$*.*

2. *For all* $\overline{p_\ell} \in [0,1]$ *such that* $\overline{p_\ell} = k\Delta$ *for* $k \in \mathbb{N}$ *do:*

   (a) *Apply the algorithm of Figure 1 with parameters* $\frac{1}{16}\epsilon$*,* $\frac{1}{4}\delta$*.*

   (b) *If* $h : X \longrightarrow \{0,1\}$ *is returned, add* $h$ *to* $\mathcal{H}$*.*

3. *Draw an unlabeled sample* $S_\ell$ *of size* $N$ *using* $D_\ell$*.*

4. *For each* $h \in \mathcal{H}$*, if* $|\{x \in S_\ell : h(x) = 1\}| < (1 - \frac{3}{16}\epsilon)|S_\ell|$ *then remove* $h$ *from* $\mathcal{H}$*.*

5. *Draw a unlabeled sample* $S$ *of size* $N$ *using* $D$*.*

6. *Let* $h' = \arg\min_{h\in\mathcal{H}} |\{x \in S : h(x) = 1\}|$*.*

7. $f_\ell(x) = h'(x)$ *for* $x \in X$*.*

Figure 2: Learning in the sense of Definition 2 using noise-tolerant PAC algorithm and unknown class priors

**Theorem 1** *Let* $\mathcal{A}^\eta$ *be a noise-tolerant algorithm as defined in Proposition 1.*

*For* $\ell \in \{0,1\}$*, the Algorithm of Figure 2 given access to* $D_\ell$ *and* $D$*, with probability at least* $1-\delta$ *outputs (in polynomial time)* $f_\ell$ *with* $\Pr_{x\sim D}(f_\ell(x) \neq t(x)) \leq \epsilon$ *for* $\ell = 1$*, and* $\Pr_{x\sim D}(f_\ell(x) \neq 1 - t(x)) \leq \epsilon$ *for* $\ell = 0$*.*

**Comment.** As a consequence we have learnability in the sense of Definition 2, since when we derive classifier $h$ from $f_1$ and $f_0$, the error of $h$ is at most the sum of the errors of $f_1$ and $f_0$. (By the error of $f_0$ we mean the probability that $f_0(x)$ is not equal to $1-t(x)$.)

***Proof.*** Let $\widehat{\Pr}_{x \in S}(\pi(x))$ denote the empirical probability that $x$ satisfies property $\pi$, with respect to sample $S$. We show first that the expression for $N$ used by the algorithm of Figure 2 guarantees that with probability at least $1 - \frac{1}{2}\delta$,

$$\forall h \in \mathcal{H} \quad \left| \Pr_{x \sim D_\ell}(h(x) = 1) - \widehat{\Pr}_{x \in S_\ell}(h(x) = 1) \right| \leq \frac{1}{16}\epsilon \tag{7}$$

$$\forall h \in \mathcal{H} \quad \left| \Pr_{x \sim D}(h(x) = 1) - \widehat{\Pr}_{x \in S}(h(x) = 1) \right| \leq \frac{1}{16}\epsilon \tag{8}$$

We are asking that the relative frequencies (over $N$ observations) of a set of at most $2|\mathcal{H}|$ events should be within $\frac{1}{16}\epsilon$ of their probabilities. Taking a union bound, it is sufficient that $N$ should satisfy: Given any $f : X \longrightarrow \{0, 1\}$, with probability at least $1 - \delta/(4|\mathcal{H}|)$

$$\left| \Pr_{x \sim D}(f(x) = 1) - \widehat{\Pr}_{x \in S}(f(x) = 1) \right| \leq \frac{1}{16}\epsilon.$$

Recall Hoeffding's inequality: Let $Y_1, \ldots, Y_N$ be Bernoulli trials with probability $p$ of success. Let $T = Y_1 + \ldots + Y_N$ denote the total number of successes. Then for $\gamma \in [0, 1]$, $\Pr(|T - pN| > \gamma N) \leq 2e^{-2N\gamma^2}$.

This means that $N$ is sufficiently large if $N$ satisfies $\delta/(4|\mathcal{H}|) \geq 2e^{-2N(\epsilon/16)^2}$. Since $|\mathcal{H}| \leq 1/\Delta$, it is sufficient for $N$ to satisfy $\delta\Delta/4 \geq 2e^{-2N(\epsilon/16)^2}$.

For $\Delta = \delta/32p(\epsilon, \delta)$,

$$\delta^2/128p(\epsilon, \delta) \geq 2e^{-2N(\epsilon/16)^2}$$

The equation is satisfied by putting $N = (16/\epsilon)^2 \cdot \log(128p(\epsilon, \delta)/\delta^2)$, polynomial in the parameters.

Assume that $\ell = 1$. We assume as before that the concept class is closed under complementation, so that the proof for $\ell = 0$ is similar but using the complement of $t$ in place of $t$.

Note that the algorithm of Figure 1 constructs a noise rate $\eta$ in the range $[0, \frac{1}{3}]$ based on $\overline{p_\ell}$, so each application of Algorithm 1 in Step (2a) uses sample size at most $p(\frac{1}{16}\epsilon^{-1}, \frac{1}{4}\delta^{-1})$. (Polynomial $p(\cdot, \cdot)$ is defined in Figure 2.)

One of the values of $\overline{p_\ell}$ used in Step (2a) as input to Algorithm 1 satisfies $|\overline{p_\ell} - p_\ell| < \Delta$. As a result, applying Proposition 1 we have that with probability $1 - \frac{1}{2}\delta$, there exists $h^* \in \mathcal{H}$ satisfying

$$\Pr_{x \sim \frac{1}{2}(D + D_1)}(h^*(x) \neq t(x)) \leq \frac{1}{16}\epsilon.$$

We may deduce that

$$\begin{array}{l} \Pr_{x \sim D_1}(h^*(x) \neq t(x)) \leq \frac{1}{8}\epsilon \\ \Pr_{x \sim D}(h^*(x) \neq t(x)) \leq \frac{1}{8}\epsilon. \end{array} \tag{9}$$

We show that with probability $1 - \frac{1}{2}\delta$

1. for $h \in \mathcal{H}$, if $\Pr_{x \sim D}(h(x) = 0 \wedge t(x) = 1) > \frac{1}{4}\epsilon$ then $h$ is eliminated in Step 4.

2. $h^*$ is not eliminated in Step 4.

3. For $h \in \mathcal{H}$, if $\Pr_{x \sim D}(h(x) = 1 \wedge t(x) = 0) > \frac{1}{2}\epsilon$ then $h$ is eliminated in Step 6.

Suppose that $\Pr_{x \sim D}(h(x) = 0 \wedge t(x) = 1) > \frac{1}{4}\epsilon$. Then $\Pr_{x \sim D_1}(h(x) = 0 \wedge t(x) = 1) > \frac{1}{4}\epsilon/p_1 \geq \frac{1}{4}\epsilon$. Hence by (7),

$$\widehat{\Pr}_{x \in S_1}(h(x) = 0 \wedge t(x) = 1) > \frac{1}{4}\epsilon - \frac{1}{16}\epsilon = \frac{3}{16}\epsilon.$$

From (7) and (9),

$$\widehat{\Pr}_{x \in S_1}(h^*(x) = 0 \wedge t(x) = 1) \leq \frac{1}{8}\epsilon + \frac{1}{16}\epsilon = \frac{3}{16}\epsilon.$$

Hence Step 4 does not eliminate $h^*$ but it eliminates all $h$ with $\Pr_{x\sim D}(h(x) = 0 \wedge t(x) = 1) > \frac{1}{4}\epsilon$.

Now suppose that $\Pr_{x\sim D}(h''(x) = 1 \wedge t(x) = 0) > \frac{1}{2}\epsilon$ for some $h'' \in \mathcal{H}$ after Step 4. We have just shown that $h''$ satisfies $\Pr_{x\sim D}(h''(x) = 0 \wedge t(x) = 1) \leq \frac{1}{4}\epsilon$. Consequently, $\Pr_{x\sim D}(h''(x) = 1) - \Pr_{x\sim D}(t(x) = 1) > \frac{1}{2}\epsilon - \frac{1}{4}\epsilon = \frac{1}{4}\epsilon$. Meanwhile note from (9) that $\Pr_{x\sim D}(h^*(x) = 1) - \Pr_{x\sim D}(t(x) = 1) \leq \frac{1}{8}\epsilon$. As a result, $\Pr_{x\sim D}(h''(x) = 1) - \Pr_{x\sim D}(h^*(x) = 1) > \frac{1}{4}\epsilon - \frac{1}{8}\epsilon = \frac{1}{8}\epsilon$. From (8),

$$\widehat{\Pr}_{x \in S}(h''(x) = 1) - \widehat{\Pr}_{x \in S}(h^*(x) = 1) > 0.$$

Hence $h''$ is eliminated at Step 6.

Hence all $h \in \mathcal{H}$ with error at least $\epsilon$ are eliminated with probability $1 - \frac{1}{2}\delta$. With probability at least $1 - \frac{1}{2}\delta$ there exists $h^* \in \mathcal{H}$ with error less that $\epsilon$. Putting these together, with probability at least $1 - \delta$ we are left with $h'$ having error less than $\epsilon$. □

# 3 Learning via Discriminant Functions without Access to $D$

We exhibit algorithms that show that learnability in the sense of Definition 1 is distinct from various well-known restrictions of PAC learnability. We also study a special case of the problem of learning monomials (in which $D$ is known to belong to a particular class of distributions), for which we have no algorithm in the distribution-independent setting.

Our algorithms are mostly proven to have the PAC property in a standard way, by arguing that the hypothesis is consistent with the data, and furthermore that it belongs to a class of hypotheses that have description length polynomial in the parameters of the problem, and sub-linear in the sample size. (This is the "Occam algorithm" property (8)). For Sections 3.2 and 3.4 we use the (more generally applicable) Vapnik-Chervonenkis dimension (9; 29) of the class of hypotheses.

## 3.1 Parity Functions

The following result distinguishes our learning setting from learnability with uniform misclassification noise, or learnability with a restricted focus of attention.

An instance is an element of $\{0,1\}^n$, representing a sequence of values of $n$ boolean variables. A *parity function* (21) 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.

**Theorem 2** *The class of parity functions is PAC learnable via discriminant functions.*

***Proof.*** Observe that the class is closed under complementation.

To learn a parity function from positive examples only, in an essentially similar way to the algorithm of (21), $\mathcal{A}_\ell$ finds the affine subspace of $GF(2)^n$ spanned by its examples, and $f_\ell$ assigns a value of 1 to elements of that subspace and a value of 0 to all other elements of the domain. (By "span" we mean with respect to $GF(2)^n$, as opposed to $\mathbb{R}^n$. $GF(2)$, the generic field with two elements 0 and 1, has addition modulo 2, so that the sum of two 0/1 vectors is their bitwise exclusive-or. Generally the positive examples of a parity function would span all of $\mathbb{R}^n$.)

In more detail, let $x_i$ be the $i$-th entry of bit vector $x$. The positive examples of a parity function satisfy $\sum_i c_i x_i = b$ (all addition and multiplication modulo 2), where $c_i, b \in GF(2)$. Let $x'$ be an arbitrary positive example; positive examples $x$ satisfy $\sum_i c_i(x_i - x'_i) = 0$, and negative examples do not satisfy this. Hence, the subspace constructed by $\mathcal{A}_1$ will be a subset of $\sum_i c_i(x_i - x'_i) = 0$, and will contain no negative examples. $\mathcal{A}_0$ constructs a subspace that contains all the negative data and no positive examples.

We have that $f_\ell(x) = 0$ for all $x$ with $t(x) = 1 - \ell$, and $f_\ell(x) = 1$ for all $x \in S_\ell$ (the unlabeled sample obtained by $\mathcal{A}_\ell$).

The overall hypothesis $h$ has description length $O(n^2)$ (a spanning set has at most $n$ vectors, each of length $n$) and $h$ is consistent with the training data; thus we have PAC-ness by the standard Occam-algorithm argument. $\square$

### 3.2 Unions of Intervals

Let $X = \mathbb{R}$, and let $t : X \longrightarrow \{0, 1\}$ be the indicator function of a union of $k$ intervals in $\mathbb{R}$. We show that the class of all such functions, is learnable by discriminant functions in time polynomial in $\epsilon^{-1}$, $\delta^{-1}$ and $k$. A union of more than one interval cannot be PAC-learned from just positive or just negative data, simply because it is impossible to guess where the data with the opposite label may lie. Learnability via discriminant functions is thus distinct from learnability from positive examples only, or from negative examples only.

**Theorem 3** *The class of unions of $k$ intervals on the real line is learnable via discriminant functions.*

***Proof.*** Construct discriminant functions $f_0$ and $f_1$ as follows. Given an (unlabeled) sample, and a point $x \in \mathbb{R}$, our discriminant function maps $x$ to the negation of its distance to its nearest neighbor in the sample. (Intuitively, it makes sense that $x$ should get a higher value if it is close to a data point in the sample.) We show furthermore that this rule creates a classifier that is "simple" (a union of $k$ intervals) and consistent with the data.

More precisely, given (unlabeled) sample $S_\ell \subset \mathbb{R}$ of size $O(k \log(\delta^{-1}\epsilon^{-1})/\epsilon)$, let $d_{NN}(x, S_\ell) = \min_{x_\ell \in S_\ell}\{|x - x_\ell|\}$. Let $f_\ell(x) = -d_{NN}(x, S_\ell)$. Recall that $h(x) = 1$ if $f_1(x) > f_0(x)$ and $h(x) = 0$ if $f_1(x) < f_0(x)$. We show that $h$ is PAC.

For $x, x' \in S_\ell$, suppose $x, x'$ belong to the same interval of $t^{-1}(\ell)$. Then $[x, x'] \subseteq h^{-1}(\ell)$, since any point between $x$ and $x'$ is closer to at least one of $x$ or $x'$ than to any point $x''$ for which $t(x'') \neq t(x)$.

Suppose $x_0 \in S_0$, $x_1 \in S_1$, $x_0 < x_1$, and there does not exist $x \in S_0 \cup S_1$ with $x_0 < x < x_1$. For a real number $x$ such that $x_0 < x < x_1$, if $x \in (x_0, \frac{1}{2}(x_0 + x_1))$ then $d_{NN}(x, S_0) < d_{NN}(x, S_1)$, so $f_0(x) > f_1(x)$ and $h(x) = 0$ for $h$ constructed according to Definition 1. Similarly, for $x \in (\frac{1}{2}(x_0 + x_1), x_1)$, $h(x) = 1$. $h(\frac{1}{2}(x_0 + x_1))$ is undefined.

For $S_0 \cup S_1$ sorted in ascending order, there are at most $2k$ pairs of consecutive points $x, x'$ in the sequence where $t(x) \neq t(x')$.

Hence $h$ is undefined on at most $2k$ elements of $\mathbb{R}$, and $h^{-1}(1)$ is a union of at most $k$ intervals, and $h^{-1}(0)$ is a union of at most $k + 1$ intervals. The VC dimension of unions of $k$ intervals is $2k$, so using the results of (8), the sample size required for PAC learning is $O(k \log(\delta^{-1}\epsilon^{-1})/\epsilon)$. $\square$

**Comment.** This nearest-neighbour rule does not work in more than one dimension, given that the input distribution $D$ is closen by an adversary. Suppose we wish to learn a linear threshold in the plane $\mathbb{R}^2$. Suppose $D$ is uniform over two parallel line segments that are very close but on opposite sides of the classification threshold. Then the probability is only about $\frac{1}{2}$ that the nearest neighbour of a data point $x$ will have the same label as $x$. In Section 3.4 we show how to learn linear thresholds in the plane using a more sophisticated rule.

### 3.3 Distinguishing the Model from the Mistake-bound Setting

In (7), Blum exhibits a concept class that is PAC learnable, but is not (in polynomial time) learnable using membership and equivalence queries, assuming that one-way functions exist. In this section we show that the concept class is PAC learnable via discriminant functions in the sense of Definition 1. We review the concept class introduced in (7). Let $X = \{0,1\}^n$.

If $A$ is a probabilistic polynomial-time algorithm that computes a function from $\{0,1\}^*$ to $\{0,1\}$, and $g$ is some function from $\{0,1\}^*$ to $\{0,1\}^*$, let $\mathcal{P}_k(A, g(s))$ denote the probability that $A(g(s)) = 1$ for strings $s$ of length $k$ generated uniformly at random.

Let $G$ be a Cryptographically Strong Pseudorandom Bit (CSB) generator with stretch $p(k) = 2k$. For polynomial $p$ a CSB generator is defined as follows.

**Definition 3** *A deterministic polynomial-time program $G$ is a CSB generator with stretch $p$ if on input $s \in \{0,1\}^k$ it produces an output in $\{0,1\}^{p(k)}$ and for all probabilistic polynomial-time algorithms $A$, for all polynomials $Q$, for sufficiently large $k$ ($k$ depending on $A$ and $Q$),*

$$|\mathcal{P}_k(A, G(s)) - \mathcal{P}_{p(k)}(A, s)| < \frac{1}{Q(k)}.$$

Thus, no polynomial-time algorithm can distinguish between strings generated uniformly at random from $\{0,1\}^{2k}$, and strings obtained by taking the output of $G$ for a random input string of length $k$. (Technically, the definition allows $A$ to be a circuit family.)

For strings $x$ and $y$, let $x \circ y$ denote their concatenation. For a bit string $x$ let $LSB[x]$ denote the rightmost bit of $x$. Let $\lambda$ denote the empty string. For bit string $s$ of length $k$, $G(s)$ is a bit string of length $2k$, and we define the following notation.

1. Let $G_0(s)$ be the leftmost $k$ bits of $G(s)$.
2. Let $G_1(s)$ be the rightmost $k$ bits of $G(s)$.
3. Let $G'_0(s)$ be the rightmost $k$ bits of $G(s)$.
4. Let $G'_1(s) = \lambda$.
5. If $i = i_1 \cdots i_d$ (where the $i_j$ are binary digits), let $G_i(s) = G_{i_d}(G_{i_{d-1}}(\cdots G_{i_1}(s)))$.

The concept class is defined as follows:

**Definition 4** *Let $k = \lfloor\sqrt{n}\rfloor - 1$ and let $\mathcal{C} = \{c_s\}_{s\in\{0,1\}^k}$ where $c_s$ is defined as follows.*

- *$c_s$ is the indicator function of* $\{x_s^i \ : \ i \in \{0,1\}^k \text{ and } \mathrm{LSB}[G_{i_1\cdots i_k}(s)] = 1\}$

*where for $i = i_1 \cdots i_k$,*

- $x_s^i = i \circ G'_{i_1}(s) \circ G'_{i_2}(G_{i_1}(s)) \circ G'_{i_3}(G_{i_1 i_2}(s)) \circ \ldots \circ G'_{i_k}(G_{i_1\cdots i_{k-1}}(s)) \circ 0^w$

*where $w$ is chosen to ensure that $|x_s^i| = n$.*

Definition 4 is slightly different from the corresponding definition of (7), where $k = \lfloor\sqrt{n}\rfloor$. We use $k = \lfloor\sqrt{n}\rfloor - 1$ so that the length of $x_s^i$ is always less that $n$, and we can then "pad it out" to a length of exactly $n$ using the string $0^w$ on the right-hand side.

Note that for any fixed $s$, a bit string of length $n$ of the form $x_s^i$ is determined entirely by $i$, its first $k$ bits. We will let the *index* of a bit string of length $n$ refer to its first $k$ bits, viewed as a binary number (to give the natural ordering on indices). For a string $x$ let $index(x)$ denote this number, regardless of whether $x$ is well-formed according to Definition 4. (If $x$ is not well-formed, $x$ is a negative example of $c_s$, i.e. $c_s(x) = 0$.) Algorithm Compute-Forward (Figure 3) shows how to take any positive example $x_s^i$, together with an index $j > i$, and construct the pair $\langle x_s^j, c_s(x_s^j)\rangle$ in polynomial time.

The following notation is used in Algorithm Compute-Forward:

1. Let $z_s^i$ be the correctly labeled example $\langle x_s^i, c_s(x_s^i)\rangle$.
2. Let $\overline{z_s^i}$ be the incorrectly labeled example $\langle x_s^i, 1 - c_s(x_s^i)\rangle$.
3. For $i_1, \ldots, i_d \in \{0,1\}^d$, let $G^{i_1 \cdots i_d}(s) = G'_{i_1}(s) \circ G'_{i_2}(G_{i_1}(s)) \circ \ldots \circ G'_{i_d}(G_{i_1 \cdots i_{d-1}}(s))$.

So $x_s^i = i \circ G^{i_1 \cdots i_k}(s) \circ 0^w$.

From (7) we know that $\mathcal{C}$ is not learnable (in time polynomial in $n$) in the mistake-bound model. We review the PAC learning algorithm of (7) and show how to adapt it to the constraint of Definition 1.

**Algorithm Compute-Forward** (7)
*On input $x_s^i$ and $j > i$,*

1. *Say $i = i_1 \cdots i_k$ and $j = j_1 \cdots j_k$. Let $r$ be the least index such that $i_r \neq j_r$. Since $j > i$ we have $i_r = 0$ and $j_r = 1$.*
2. *Extract from $x_s^i$ the portions:*

$$\begin{aligned} u &= G'_{i_1}(s) \circ G'_{i_2}(G_{i_1}(s)) \circ G'_{i_3}(G_{i_1 i_2}(s)) \circ \ldots \circ G'_{i_{r-1}}(G_{i_1 \cdots i_{r-2}}(s)) \\ &= G^{i_1 \cdots i_{r-1}}(s). \\ v &= G'_{i_r}(G_{i_1 \cdots i_{r-1}}(s)) = G_{j_r}(G_{j_1 \cdots j_{r-1}}(s)). \end{aligned}$$

3. *Notice that $G'_{j_r}(G_{j_1 \cdots j_{r-1}}(s)) = \lambda$. Since $v = G_{j_r}(G_{j_1 \cdots j_{r-1}}(s))$, we can use $v$ as an intermediate point in the computation of those parts of $z_s^j$ that differ from $z_s^i$.*
4. *If $r = k$, output: $\langle j \circ u \circ \lambda, \mathrm{LSB}[v]\rangle$. Otherwise, output: $\langle j \circ u \circ \lambda \circ G^{j_{r+1} \cdots j_k}(v), \mathrm{LSB}[G_{j_k \cdots j_{r+1}}(v)]\rangle$.*

Figure 3: Algorithm from (7)

We noted that Algorithm Compute-Forward, given a positive example $x_s^i$ and $j > i$, produces a correctly-labeled example $\langle x_s^j, c_s(x_s^j)\rangle$. Based on this observation, we assign values to examples as shown in Figure 4.

**Theorem 4** *The concept class of (7) is learnable via discriminant functions.*

***Proof.*** We use the algorithm of Figure 4 to construct discriminant functions.

Recall that for $x \in \{0,1\}^n$, $index(x)$ denotes the $k$ bit binary number forming a prefix of $x$, for $k = \lfloor \sqrt{n} \rfloor - 1$. For $\ell \in \{0,1\}$, $\mathcal{A}_\ell$ denotes the instance of the algorithm that is given access to $D_\ell$.

*Input $S_\ell$, a sample of unlabeled elements of $X = \{0,1\}^n$.*

1. *Apply algorithm Consistency-Check to $S_\ell$; if $S_\ell$ fails the test, then for all $x \in X$, $f_\ell(x) = \frac{1}{2}$. Else:*

2. *Let $m$ and $M$ be the minimum and maximum indices of elements of $S_\ell$. Call these elements $x_s^m$ and $x_s^M$ respectively; since Consistency-Check has been passed, they are unique. For $x \in X$, $index(x) = j$, if $j > M$ or $j < m$ then $f_\ell(x) = 0$.*

3. *If $x \in S_\ell$ then $f_\ell(x) = 1$.*

4. *Otherwise, if $\langle x_s^j, 1\rangle$ =Compute-Forward($x_s^m, j$) and furthermore, $\langle x_s^M, 1\rangle$ =Compute-Forward($x_s^j, M$) then let $f_\ell(x_s^j) = 1$.*

5. *Otherwise, let $f_\ell(x_s^m) = 0$.*

**Algorithm Consistency-Check**

1. *If there exist $x_1, x_2 \in S_\ell$ with $x_1 \neq x_2$, but $index(x_1) = index(x_2)$, then fail.*

2. *If there exist $x_1, x_2 \in S_\ell$ such that $index(x_2) > index(x_1)$, yet with Compute-Forward($x_1, index(x_2)$)$\neq \langle x_2, 1\rangle$, then fail.*

Figure 4: Assigning values to unlabeled data for concept class of (7)

As in (7), we will argue that what we have is an "Occam Algorithm" in the sense of Blumer et al. (8) which is consistent with the training data. Specifically, $\mathcal{A}_1$ and $\mathcal{A}_0$ memorize at most 2 training examples each ($\mathcal{A}_0$ possibly memorizes none) and their combined hypothesis (the $h$ in Definition 1) is consistent with the training data.

In particular, $\mathcal{A}_1$ (and possibly also $\mathcal{A}_0$) just retains $x_s^m$ and $x_s^M$, since for any unlabeled $x$, the label assigned to it is computed (in polynomial time) using $x_s^m$ and $x_s^M$. (In the case of $\mathcal{A}_0$, the sample $S_0$ may fail the test *Consistency-Check*, in which case no examples are memorized.) Hence the description length of the rule that labels examples, is $O(n)$.

Note that $f_1$ from $\mathcal{A}_1$ will now give a value of 1 to any positive example whose index is between the largest and smallest indices it has seen so far, and will give value of 0 to all other examples. If an example $x \in X$ is either negative or is ill-formed ("bad" in the terminology of (7)), then Step 4 will ensure $f_1(x) = 0$, even if $index(x)$ is between $m$ and $M$.

At the same time, we claim that $f_0$ from $\mathcal{A}_0$ gives a value of $\leq \frac{1}{2}$ to all positive examples. Suppose for a contradiction that $\mathcal{A}_0$ gives a value of 1 to positive example $x_s^j$. Then $\mathcal{A}_0$ must have in its collection an unlabeled example $x_s^M$ and $x_s^j$ must predict $x_s^M$ as being positive. But that implies that $x_s^M$ must be positive, and since it belongs to $S_0$ it is negative, a contradiction.

$\mathcal{A}_0$ ensures that $f_0(x) \geq \frac{1}{2}$ for all $x \in S_0$. $\mathcal{A}_1$ ensures that $f_1(x) \geq 1$ for all $x \in S_1$ and $f_1(x) = 0$ for all negative $x$ (including all $x \in S_0$). Hence the combined classifier $h$ is consistent with the training data. $\square$

### 3.4 Linear Separators in the Plane

For $X = \mathbb{R}^2$, suppose each $x \in X$ is labeled 0 or 1 according to whether its coordinates satisfy some linear inequality; that is, a concept is a half-space in $\mathbb{R}^2$. This problem is well-known to be PAC-learnable in the standard setting; generally for $X = \mathbb{R}^n$ it reduces to linear programming.

Given a sample $S_\ell$ of points in $t^{-1}(\ell)$, note that points within their convex hull[2] ought to receive a "high" value from $f_\ell$, since the convex hull must be a subset of $t^{-1}(\ell)$. We need to be able to deal with the case when the convex hull has most or all of $S_\ell$ at its vertices, as would happen for an input distribution $D_\ell$ 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 value to points inside $P$. Then give an intermediate value 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 given the correct label.

1. *Draw a sample $S_\ell$ of size $N = \Theta(\log(1/\delta\epsilon)/\epsilon^2)$.*
2. *Let polygon $P_\ell$ be the convex hull of $S_\ell$.*
3. *Let $Q_\ell$ be a polygon having at most $\lceil\sqrt{N}\rceil$ edges such that*
   (a) *Every edge of $Q_\ell$ intersects $P_\ell$ at a single vertex, and*
   (b) *Adjacent edges of $Q_\ell$ contain vertices of $P_\ell$ that are at most $\sqrt{N}$ apart in the adjacency sequence of $P_\ell$'s vertices.*
4. *Define discriminant function $f_\ell$ as follows.*
   (a) *For all $x$ in the interior or boundary of $P_\ell$, $f_\ell(x) = 1$.*
   (b) *For each connected region $R$ in $Q_\ell \setminus P_\ell$ let $A(R)$ denote its area. For $x \in R$ let $f_\ell(x) = (A(R)+1)^{-1}$. If $A(R)$ is infinite let $f_\ell(x) = 0$.*
   (c) *For $x \notin Q_\ell$ let $f_\ell(x) = -1$.*

Figure 5: Assigning values to unlabeled data for linear separators in the plane

**Theorem 5** *Linear separators in the plane are learnable via discriminant functions.*

***Proof.*** Figure 5 shows the algorithm we use to construct discriminant functions; it is not hard to check that the steps can be carried out in polynomial time. Figure 6 illustrates the construction on an example.

Let $h : \mathbb{R}^2 \longrightarrow \{0,1\}$ be the hypothesis constructed from $f_0$ and $f_1$. We show below that for $\ell \in \{0,1\}$, $h^{-1}(\ell)$ is a region bounded by $O(\sqrt{N})$ line segments. We also show that $h$ is consistent with the data, i.e. that for $x \in S_\ell$ (the unlabeled sample drawn by $\mathcal{A}_\ell$), we have $h(x) = \ell$. As before, PAC-ness follows from an Occam-algorithm argument; the class of hypotheses has VC dimension $O(\sqrt{N})$, sublinear in the sample size.

[2]The *convex hull* of a finite set $S$ of points is the smallest convex polygon (more generally, polytope) that contains $S$. Any vertex of the convex hull of $S$ is a member of $S$.

To show consistency of the hypothesis, suppose $x \in S_1$, i.e. $x$ is a positive example. Then $f_1(x) = 1$ since $x$ lies in the interior or on the boundary of $P_1$ (rule 4a). By contrast, when $f_0$ is constructed, $x$ lies strictly outside the convex hull of the negative data, so either rule 4b or 4c is applied, giving $f_0(x)$ a value less than 1. By symmetry, members of $S_0$ are also correctly labeled.

Next we prove our claim that the boundary between the points labeled 0 by $h$, and the points labeled 1, is indeed simple. (Specifically, $h^{-1}(0)$ and $h^{-1}(1)$ are bounded by $O(\sqrt{N})$ line segments.) Let $L$ be the line that defines the target linear threshold function $t$. Let $\mathcal{R}_\ell$ be the set of connected regions constructed by $\mathcal{A}_\ell$ that lie between $P_\ell$ and $Q_\ell$. For $R \subseteq \mathbb{R}^2$ let $CH(R)$ denote the convex hull of $R$. Observe that

1. no straight line may pass through more than 2 elements of $\mathcal{R}_\ell$. (If that occurred, suppose the line passes through $R, R', R'' \in \mathcal{R}_\ell$ in that order. Note that $P_\ell \cup R \cup R''$ is convex. That makes it impossible for the line to cut $R'$, which is outside $P_\ell \cup R \cup R''$.)

2. at most one element of $\mathcal{R}_0$ (respectively, $\mathcal{R}_1$) may intersect $L$. (If two of them intersected $L$, there would be an edge of $Q_\ell$ on the opposite side of $L$ from $P_\ell$, hence a vertex of $P_\ell$ on the wrong side of $L$.)

3. for $R \in \mathcal{R}_\ell$, $CH(R)$ is a region bounded by three line segments. ($R$ has two "outer" edges and a concave sequence of edges from $P_\ell$ connecting them.)

Suppose that $R_0 \in \mathcal{R}_0$ intersects $R_1 \in \mathcal{R}_1$. Then $CH(R_0)$ intersects $CH(R_1)$. From Observation 3 above, the boundary of $CH(R_0)$ has only 2 line segments on the opposite side of $L$ from $P_0$, and from Observation 1 the boundary of $CH(R_0)$ intersects at most 4 elements of $\mathcal{R}_1$. For all remaining regions $R'_1 \in \mathcal{R}_1$, either $R'_1 \subset R_0$ (so that for $x \in R'_1$, $f_1(x) > f_0(x)$) or $R'_1 \cap R_0 = \emptyset$ (so that again, for $x \in R'_1$, $f_1(x) > f_0(x)$).

For $\ell \in \{0, 1\}$ let $P'_\ell = P_\ell \cup \{R_\ell \ : \ h(R_\ell) = \ell\}$. Note that $P'_\ell \subseteq h^{-1}(\ell)$ and has at most $3\lceil\sqrt{N}\rceil$ edges.

The portion of $t^{-1}(0)$ not in $P'_0 \cup P'_1$ is divided into $O(\sqrt{N})$ regions by the remaining edges of $Q_0$ and the two edges of $Q_1$ that intersect $t^{-1}(0)$. $h$ is constant within each of these regions, which allows us to deduce that $h^{-1}(0)$ is indeed bounded by a set of line segments of size $O(\sqrt{N})$. By a similar argument, $h^{-1}(1)$ is bounded by $O(\sqrt{N})$ lines. □

### 3.5 Monomials over Attribute Vectors having a Product Distribution.

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 under the restriction of Definition 1, even for monotone (negation-free) monomials. If $\mathcal{A}_0$ and $\mathcal{A}_1$ are allowed to be different algorithms ($\mathcal{A}$ is allowed to treat the positive data differently from the negative data), 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). $f_0$ from $\mathcal{A}_0$ assigns a value of $\frac{1}{2}$ to all boolean vectors. $\mathcal{A}_1$ uses its data to find a PAC hypothesis, and assigns a value of 1 to examples satisfying that hypothesis, and 0 to other examples.

The following problem arises when $\mathcal{A}$ is oblivious to whether it is receiving the positive data. The distribution over the negative examples could in fact produce boolean vectors that satisfy some monomial $f$ that differs from target monomial $t$, but if $D(f^{-1}(1) \cap t^{-1}(1)) > \epsilon$ this may give excessive error.

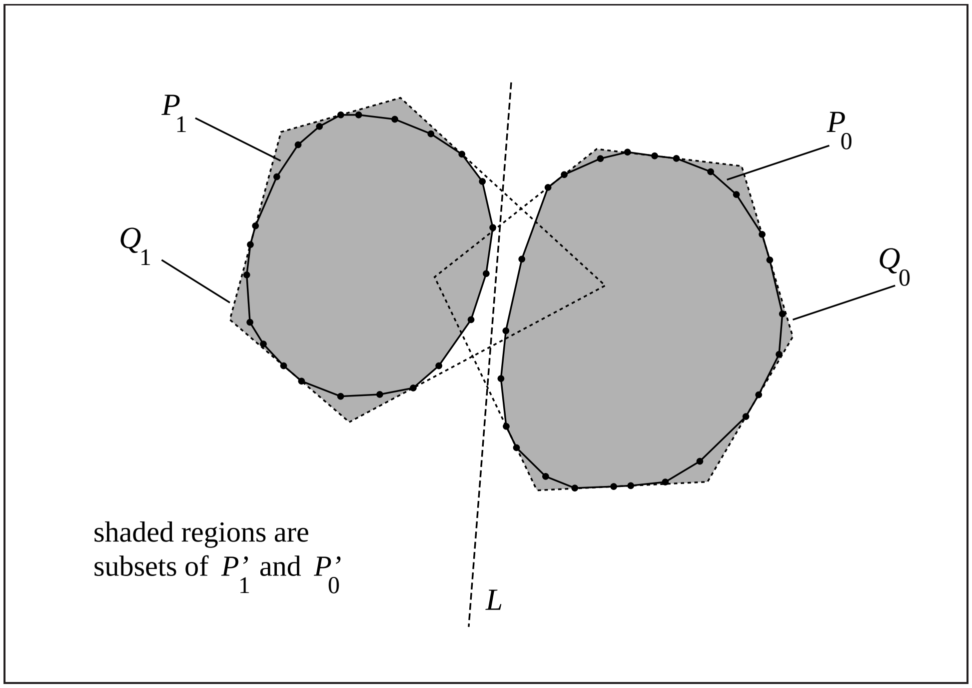


Figure 6: Illustration of algorithm for learning linear separators in two dimensions

In view of the importance of the concept class of monomials, 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 1 the problem would be solved since monomials are learnable in the presence of uniform misclassification noise (2)) and the distribution-independent setting.

1. *Draw a sample $S_\ell$ of size $N = \tilde{O}((n^3/\epsilon)^2 \log(\frac{1}{\delta}))$.*
2. *For $x \in X$ let $\widehat{\psi}_\ell^j(x)$ denote the fraction of elements of $S_\ell$ whose $j$-th entry is equal to the $j$-th entry of $x$.*
3. *For $x \in X$, $f_\ell(x) = \Pi_{j=1}^n \widehat{\psi}_\ell^j(x)$.*

Figure 7: Algorithm for learning monomials

**Theorem 6** *Monomials over the boolean domain are learnable via discriminant functions, provided that the input distribution $D$ is known to be a product distribution.*

**Comments.** We use the algorithm of Figure 7 which simply fits a product distribution to its data and assigns a value to unlabeled vector $x$ that is the estimated likelihood of $x$. The proof that it works heavily exploits the assumption that $D$ is a product distribution, and does not appear to extend to larger class of distributions (for example, mixtures of product distributions (13; 17)) or more general classes of boolean functions.

***Proof.*** We show that the algorithm given in Figure 7 constructs discriminant functions which, when combined to get $h$ according to Definition 1, ensure that $h$ is PAC.

For $x \sim D$, $x = x_1 x_2 \dots x_n$ where $x_j$ is a 0/1 random variable which is independent of $x_k$ for $k \neq j$. By a *relevant attribute* of $t$ we mean any $x_j$ whose value is fixed for all $x$ that satisfy $t$. Let $t_j$ denote that value. Let $\mathcal{R}$ denote the set of relevant attributes and let $\mathcal{I}$ denote the remaining (irrelevant) attributes.

We say that an example $x' = x'_1 x'_2 \dots x'_n$ with $t(x') = \ell$ is *ordinary* if for all $b \in \{0,1\}$ and $j \in \{1,\dots,n\}$ such that $\Pr_{x\sim D_\ell}(x_j = b) > 1 - \frac{\epsilon}{n}$, we have $x'_j = b$. (Thus, an ordinary example is one that does not have any "very unusual" attribute values in comparison with random examples having the same label. If there happen to be no bit positions that are very "reliable" for random examples with the same label, then the property becomes vacuous, or true for all bit strings.)

Note that for $\ell \in \{0,1\}$, $\Pr_{x\sim D_\ell}(x \text{ is ordinary}) \geq 1 - \epsilon$. Consequently, $\Pr_{x\sim D}(x \text{ is ordinary}) \geq 1 - \epsilon$. We will show that with probability at least $1 - \delta$, all ordinary examples end up correctly labeled.

Let $\widehat{\Pr}_{x\in S_\ell}(x_j = b)$ denote the empirical probability that $x_j = b$, and we show that sample size $N$ is large enough to ensure that with probability $1 - \delta$, for $b \in \{0,1\}$, $j \in \{1,\dots,n\}$,

$$|\widehat{\Pr}_{x\in S_\ell}(x_j = b) - \Pr_{x\sim D_\ell}(x_j = b)| \leq \frac{\epsilon}{8n^3}. \tag{10}$$

Applying the same Hoeffding bound as in Theorem 1, it is sufficient that $N$ should satisfy $2e^{-2N(\epsilon/8n^3)^2} \leq \frac{\delta}{2n^2}$ which is satisfied by $N$ as prescribed in Figure 7.

For $\ell \in \{0,1\}$, $x \in X$ let $\psi_\ell^j(x)$ denote the probability that a random vector with label $\ell$ agrees with $x$ on the $j$-th entry. Note that if $t(x) = \ell$ then $\widehat{\psi}_\ell^j(x)$ (as defined in the algorithm of Figure 7) is an empirical estimate of $\psi_\ell^j(x)$.

Let $\psi_\ell(x) = \Pi_j \psi_\ell^j(x)$. Note that $f_\ell(x)$ is an estimate of $\psi_\ell(x)$ (in the sense that $f_\ell(x)$ converges to $\psi_\ell(x)$ as the sample size increases). We know from (10) that

$$\widehat{\psi}_\ell^j(x) \in \Big[\psi_\ell^j(x) - \frac{\epsilon}{8n^3}, \psi_\ell^j(x) + \frac{\epsilon}{8n^3}\Big].$$

If $\psi_\ell^j(x) > \epsilon/n$, then

$$\widehat{\psi}_\ell^j(x)/\psi_\ell^j(x) \in \Big[1 - \frac{1}{8n^2}, 1 + \frac{1}{8n^2}\Big]. \tag{11}$$

Suppose $x$ is ordinary and negative. Observe that $f_1(x) = 0$. (This is because $x$ must have an attribute value that disagrees with all corresponding attribute values in the positive data.) Furthermore, Equation (11) holds for $\ell = 0$ and all $j$, implying that

$$\widehat{\psi}_0(x)/\psi_0(x) \in \Big[1 - \frac{1}{4n}, 1 + \frac{1}{4n}\Big]. \tag{12}$$

So with probability $1 - \delta$, $f_0(x) > 0$, since $f_0(x) = 0$ would contradict Equation (12) taken with the observation that $\psi_0(x) > 0$. Hence with probability $1 - \delta$, all ordinary negative examples are correctly labeled.

Suppose $x$ is ordinary and positive. We will show that with probability $1 - \delta$, $f_1(x)/f_0(x) > 1$ for all ordinary positive examples. Observe that for $j \in \mathcal{R}$, $\psi_1^j(x) = \widehat{\psi}_1^j(x) = 1$. (This is because all positive examples must agree on all the relevant attributes.) We have

$$\begin{aligned} f_1(x) &= \Pi_{j\in\mathcal{I}} \widehat{\psi}_1^j(x) \\ f_0(x) &= \Pi_{j\in\mathcal{I}} \widehat{\psi}_0^j(x) \Pi_{j\in\mathcal{R}} \widehat{\psi}_0^j(x). \end{aligned}$$

For $j \in \mathcal{I}$, $\psi_1^j(x) = \psi_0^j(x)$ (for a product distribution $D$, the value of an irrelevant attribute is selected independently of the label class of a bit string). Hence Equation (11) applies for $\ell = 0$ or $1$, $j \in \mathcal{I}$.

$$\frac{f_1(x)}{f_0(x)} \geq \frac{(1-(1/8n^2))^n}{(1+(1/8n^2))^n}\Big(\frac{1}{\Pi_{j\in\mathcal{R}}\widehat{\psi}_0^j(x)}\Big).$$

There exists $j^* \in \mathcal{R}$ such that for a fraction at least $\frac{1}{n}$ of elements $x' \in S_0$, $x'_{j^*} \neq x_{j^*}$. (Each negative example must disagree with $x$ on at least one relevant attribute.) Hence,

$$\begin{array}{rcl} \widehat{\psi}_0^{j^*}(x) & \leq & 1-(1/n) \\ \psi_0^{j^*}(x) & \leq & 1-(1/n)+(\epsilon/8n^2) < 1-(1/2n). \end{array}$$

Hence

$$\frac{f_1(x)}{f_0(x)} \geq \frac{(1-(1/8n^2))^n}{(1+(1/8n^2))^n}\Big(\frac{1}{1-(1/2n)}\Big) > 1,$$

as required. Hence with probability $1-\delta$, all ordinary positive examples are correctly labeled. $\square$

# 4 Conclusion and Open Problems

The algorithms we have given differ significantly from previous PAC algorithms, which usually work by minimizing the empirical error rate, and arguing that the way a hypothesis is constructed ensures that the true error is close to the empirical error. The constraint that we expressed in Definition 1 forces the positive data and the negative data to be processed independently — an algorithm does not have access to the empirical error.

This lack of access to the empirical error appears to be quite a severe constraint, one that might render certain learning problems intractable in the context of PAC learning. Indeed, we have so far failed to find an algorithm in this setting which learns monomials over the boolean domain, assuming no knowledge of the input distribution. We have also not obtained an algorithm for learning linear threshold functions in more than two dimensions. Despite those limitations, our positive results have distinguished learnability subject to this constraint from various other constraints on PAC learnability that have been studied in the past.

Clearly, the main open question raised by this paper is to elucidate the relationship between learnability via discriminant functions (Definition 1), and basic PAC learnability. Furthermore, if they are not equivalent, can they be distinguished using a well-known learning problem, such as monomials over the boolean domain?

We have a relatively good understanding of learnability subject to the slightly less severe constraint of Definition 2. Namely, it is intermediate between learnability with uniform misclassification noise, and standard PAC learnability. Furthermore, subject to the Noisy Parity Assumption (that it is hard to learn parity functions in the presence of random misclassification noise given the uniform distribution over input vectors) it is strictly a less severe constraint that learnability with uniform misclassification noise, since we have shown (Section 3.1) how to learn parity functions using the more severe constraint of Definition 1.

# References

[1] E.L. Allwein, R.E. Schapire and Y. Singer. (2000). Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers. *Journal of Machine Learning Research*, 1, pp. 113-141.

[2] D. Angluin and P. Laird. (1988). Learning from noisy examples. *Machine Learning*, 2(4), pp. 343-370.

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

[4] S. Ben-David and E. Dichterman. (1998). Learning with Restricted Focus of Attention. *J. of Computer and System Sciences*, 56(3), pp. 277-298.

[5] S. Ben-David and E. Dichterman. (1994). Learnability with restricted focus of attention guarantees noise-tolerance. *5th International Workshop on Algorithmic Learning Theory*, pp. 248-259.

[6] C.M. Bishop. (1995). *Neural Networks for Pattern Recognition.* Oxford University Press.

[7] A. Blum. (1994). Separating Distribution-Free and Mistake-Bound Learning Models over the Boolean Domain. *SIAM J. Comput.* 23(5), pp. 990-1000.

[8] A. Blumer, A. Ehrenfeucht, D. Haussler and M.K. Warmuth. (1987). Occam's razor. *Information Processing Letters*, 24, pp. 377-380.

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

[10] 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.

[11] 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.

[12] N. Cristianini and J. Shawe-Taylor. (2000). *An Introduction to Support Vector Machines.* Cambridge University Press.

[13] M. Cryan, L.A. Goldberg and P.W. Goldberg. (2001). Evolutionary Trees can be Learned in Polynomial Time in the Two-State General Markov Model. *SIAM J. Comput.* 31(2), pp. 375-397.

[14] S. Dasgupta. (1999). Learning mixtures of Gaussians. *40th IEEE Symposium on Foundations of Computer Science*, pp. 634-644.

[15] F. Denis. (1998). PAC Learning from Positive Statistical Queries. *Procs. of the 9th International Conference on Algorithmic Learning Theory*, LNAI 1501, pp. 112-126.

[16] R.O. Duda and P.E. Hart. (1973). *Pattern Classification and Scene Analysis.* Wiley, New York.

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

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

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

[20] 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.

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

[22] M.J. Kearns. (1998). Efficient Noise-Tolerant Learning From Statistical Queries. *Journal of the ACM*, 45, pp. 983-1006.

[23] 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.

[24] M.J. Kearns and R.E. Schapire. (1994). Efficient Distribution-free Learning of Probabilistic Concepts. *Journal of Computer and System Sciences*, 48(3) pp. 464-497.

[25] F. Letouzey, F. Denis and R. Gilleron. (2000). Learning from Positive and Unlabeled Examples. *Procs. of the 11th International Conference on Algorithmic Learning Theory.* pp. 71-85.

[26] N. Palmer and P.W. Goldberg. (2005). PAC Classification based on PAC Estimates of Label Class Distributions. *University of Warwick, CS Dept Tech. Rept.* No. 411.

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

[28] R. Schapire. (1990). The Strength of Weak Learnability. *Machine Learning*, 5, pp. 197-227.

[29] V.N. Vapnik. (2000). *The Nature of Statistical Learning Theory* second edition, Springer-Verlag, New York.