On PAC Learning Using Winnow, Perceptron, and a Perceptron-Like Algorithm

Rocco A. Servedio*
Division of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138
rocco@deas.harvard.edu

Abstract

In this paper we analyze the PAC learning abilities of several simple iterative algorithms for learning linear threshold functions, obtaining both positive and negative results. We show that Littlestone's Winnow algorithm is not an efficient PAC learning algorithm for the class of positive linear threshold functions. We also prove that the Perceptron algorithm cannot efficiently learn the unrestricted class of linear threshold functions even under the uniform distribution on boolean examples. However, we show that the Perceptron algorithm can efficiently PAC learn the class of nested functions (a concept class known to be hard for Perceptron under arbitrary distributions) under the uniform distribution on boolean examples. Finally, we give a very simple Perceptron-like algorithm for learning origin-centered halfspaces under the uniform distribution on the unit sphere in R^n. Unlike the Perceptron algorithm, which cannot learn in the presence of classification noise, the new algorithm can learn in the presence of monotonic noise (a generalization of classification noise). The new algorithm is significantly faster than previous algorithms in both the classification and monotonic noise settings.

1 INTRODUCTION

Learning an unknown linear threshold function from labelled examples is one of the oldest and most thoroughly studied problems in machine learning. Many different versions of and algorithms for this problem have been proposed and analyzed in the four decades since the Perceptron algorithm [37] was introduced. Even today, learning problems involving linear threshold functions continue to be the subject of intensive research in both the applied and theoretical machine learning communities.

The classical Perceptron algorithm [37] and Littlestone's Winnow algorithm [25, 26] are two algorithms for learning linear threshold functions which have been studied extensively in the on-line mistake-bound model [6, 15, 24, 27, 30, 32, 42]. In this model the learning algorithm sequentially makes predictions on examples as they are received, using a hypothesis which it can update after each example, and the performance of an algorithm is measured by the worst-case number of mistakes it makes on any example sequence. Despite widespread interest in the Perceptron and Winnow algorithms, relatively little is known about their performance in Valiant's commonly used Probably Approximately Correct, or PAC, model of concept learning [41]. In this paper we establish positive and negative results on the PAC learning abilities of Winnow, Perceptron, and a related algorithm, thus helping to clarify their status in the PAC model.

In Valiant's model there is a fixed distribution $\mathcal{D}$ from which labelled examples are drawn; the goal of the learner is to find a hypothesis which closely approximates the target function under distribution $\mathcal{D}$. It has long been known that if the space of possible examples is the n-dimensional unit sphere S^{n-1}, then the Perceptron algorithm is not an efficient PAC learning algorithm for the class of linear threshold functions because of "hard" distributions which concentrate their weight on examples lying close to the separating hyperplane. Baum [7] has shown that if $\mathcal{D}$ is restricted to be the uniform distribution on S^{n-1}, though, then Perceptron is an efficient PAC learning algorithm for the class of linear threshold functions. Schmitt [38] has recently shown that Perceptron is not an efficient PAC learning algorithm for the class of boolean threshold functions. His proof works by exhibiting a nested boolean function and a distribution which is concentrated on "hard" examples for that function. No results analogous to these have appeared for the Winnow algorithm; as Schmitt has noted, it was not known "whether Littlestone's rules can PAC identify in polynomial time" [38].

(We note that various generic techniques exist [1, 19, 25] for converting from mistake-bound algorithms to algorithms which satisfy PAC criteria. In each of these conversions the running time in the PAC model depends on the mistake bound of the original algorithm. However, since the best mistake bounds known for the Perceptron and Winnow algo-

*Supported in part by an NSF Graduate Fellowship, by NSF grant CCR-95-04436 and by ONR grant N00014-96-1-0550.

rithms on nested functions (or general linear threshold functions) are exponentially large [18] [25], these generic techniques only provide an exponential upper bound on the running time of Perceptron and Winnow in the PAC model, and cannot be used to say anything about whether or not Perceptron and Winnow are *efficient* PAC learning algorithms for these concept classes.)

This brings us to the contributions of the present paper. In Section 3 we answer Schmitt's question by proving that the Winnow algorithm is not an efficient PAC learning algorithm for the class of positive linear threshold functions. To the best of our knowledge this is the first negative result for Winnow in the PAC model. In Section 4, we strengthen Schmitt's negative result for Perceptron by giving a simple proof that the Perceptron is not an efficient PAC algorithm for the class of linear threshold functions under the uniform distribution on boolean examples. This provides an interesting contrast to Baum's positive uniform-distribution result for the unit sphere. In Section 5 we show that under the uniform distribution on boolean examples, the Perceptron algorithm is an efficient PAC algorithm for the class of nested boolean functions. This suggests that Schmitt's negative result for Perceptron on nested functions depends on choosing an adversarial distribution. Finally, in Section 6 we analyze an extremely simple algorithm for PAC learning origin-centered halfspaces (linear threshold functions with threshold 0) under the uniform distribution on S^{n-1}. We show that the new algorithm is highly noise-tolerant and is significantly faster than several previous algorithms which have been considered for this problem.

2 PRELIMINARIES

A *concept class* C on an *example space* X is a collection of boolean functions on X. The sets C and X are striated, i.e. $C = \cup_{n \geq 1} C_n$ and $X = \cup_{n \geq 1} X_n$; throughout this paper X_n will be either the boolean cube $\{0,1\}^n$ or the unit n-sphere S^{n-1}. A boolean function $f : X_n \to \{-1, 1\}$ is a *threshold function* if there is a weight vector $w \in R^n$ and a threshold $\theta \in R$ such that $f(x) = 1$ iff $w \cdot x \geq \theta$. Such a pair (w, θ) is said to *represent* f. See [14], [33] for an extensive treatment of boolean threshold functions.

2.1 PERCEPTRON

Throughout its execution the Perceptron algorithm maintains a weight vector w and a threshold θ as its current hypothesis. The algorithm proceeds in a series of steps; in each step it receives an example x, uses its hypothesis to make a prediction on x, and adjusts its hypothesis if the prediction is incorrect. Initially the algorithm starts with weight vector $w = (0, \ldots, 0)$ and threshold $\theta = 0$. Upon receiving an example x, the algorithm predicts according to the threshold function $w \cdot x \geq \theta$. If the prediction is incorrect, the hypothesis is updated according to the following rule:

- On a false positive prediction, set $w \leftarrow w - x$ and set $\theta \leftarrow \theta + 1$.
- On a false negative prediction, set $w \leftarrow w + x$ and set $\theta \leftarrow \theta - 1$.

No change is made if the hypothesis was correct on x. If each example x belongs to $\{0,1\}^n$, then each w_i and θ will always be integers; this fact will prove useful later.

2.2 WINNOW

The Winnow algorithm takes as input an initial vector $w^I \in R^n$, a promotion factor $\alpha \in R$, and a threshold $\theta \in R$. The algorithm requires that w^I is positive (i.e., each coordinate of w^I is positive), that $\alpha > 1$, and that $\theta > 0$. Like the Perceptron algorithm, Winnow proceeds in a series of trials and predicts in each trial according to the threshold function $w \cdot x \geq \theta$. If the prediction is correct, then no update is performed; otherwise the weights are updated as follows:

- On a false positive prediction, for all i set $w_i \leftarrow \alpha^{-x_i} w_i$;
- On a false negative prediction, for all i set $w_i \leftarrow \alpha^{x_i} w_i$.

It should be noted that in this form, Winnow is only capable of expressing positive threshold functions as its hypotheses. This limitation can be easily overcome, however, by using various simple transformations on the input (see [25], [26]).

2.3 PAC LEARNING USING ON-LINE LEARNING ALGORITHMS

In Valiant's PAC learning model [41], the learning algorithm has access to an *example oracle* $EX(c, \mathcal{D})$ which, in one time step, provides a labelled example $\langle x, c(x) \rangle$ where x is drawn from the distribution $\mathcal{D}$ on the example space X. The function $c \in C$ is called the *target concept*; the goal of the learning algorithm is to construct a hypothesis h which, with high probability, has low error with respect to c. We thus have the following:

Definition 1 *We say that an on-line learning algorithm (such as Winnow or Perceptron) is an* efficient PAC learning algorithm for concept class C over $\{0,1\}^n$ *if there is a polynomial $p(\cdot, \cdot, \cdot)$ such that the following conditions hold for any $n \geq 1$, any distribution $\mathcal{D}$ over $\{0,1\}^n$, any $c \in C_n$, and any $0 < \epsilon, \delta < 1$:*

- *Given any example $x \in \{0,1\}^n$, algorithm A always evaluates its hypothesis on x and (once provided with $c(x)$) updates its hypothesis in poly(n) time;*
- *if algorithm A is run on a sequence of examples generated by successive calls to $EX(c, \mathcal{D})$, then with probability at least $1 - \delta$ algorithm A will generate a hypothesis h such that $\Pr_{x \in \mathcal{D}}[h(x) \neq c(x)] < \epsilon$ after at most $p(n, \frac{1}{\epsilon}, \frac{1}{\delta})$ calls to $EX(c, \mathcal{D})$.*

An algorithm A is an efficient PAC learning algorithm under distribution $\mathcal{D}$ if it satisfies the above definition for a fixed distribution $\mathcal{D}$.

2.4 NESTED FUNCTIONS

Several of the results in this paper involve nested functions. This class was introduced by Anthony, Brightwell and Shawe-Taylor in [4].

Definition 2 *The class of nested functions over $x_1, \ldots, x_n$, denoted NF_n, is defined as follows:*

1. *for $n = 1$, the functions x_1 and $\overline{x}_1$ are nested.*
2. *for $n > 1$, $f(x_1, \ldots, x_n)$ is nested if $f = g * l_n$, where g is a nested function on $x_1, \ldots, x_{n-1}$, $*$ is either $\vee$ or $\wedge$, and l_n is either x_n or $\overline{x}_n$.*

It is easy to verify that the class of nested functions is equivalent to the class of 1-decision lists of length n in which the variables appear in the reverse order $x_n, \ldots, x_1$. The following lemma establishes a canonical representation of nested functions as threshold functions:

Lemma 1 *Any $f \in NF_n$ can be represented by a boolean threshold function*

$$w_1x_1 + \cdots + w_nx_n \geq \theta_n,$$

with $\theta_n = k + \frac{1}{2}$ for some integer $k, w_i = \pm 2^{i-1}$, and $\sum_{w_i<0} w_i < \theta_n < \sum_{w_i>0} w_i$.

Proof: The proof is by induction on n. For $n = 1$ the appropriate threshold function is $x_1 \geq \frac{1}{2}$ or $-x_1 \geq -\frac{1}{2}$. For $n > 1$, f must be of the form $g * l_n$, where g is a nested function on $x_1, \ldots, x_{n-1}$. By the induction hypothesis, g can be expressed as a threshold function $w_1x_1 + \cdots + w_{n-1}x_{n-1} \geq \theta_{n-1}$, with $w_1, \ldots, w_{n-1}, \theta_{n-1}$ satisfying the conditions of the lemma. There are four possibilities:

1. $f = g \wedge x_n$: then f is
$$w_1x_1 + \ldots + w_{n-1}x_{n-1} + 2^{n-1}x_n \geq \theta_n = \theta_{n-1} + 2^{n-1}$$
2. $f = g \wedge \overline{x}_n$: then f is
$$w_1x_1 + \ldots + w_{n-1}x_{n-1} - 2^{n-1}x_n \geq \theta_n = \theta_{n-1}$$
3. $f = g \vee x_n$: then f is
$$w_1x_1 + \ldots + w_{n-1}x_{n-1} + 2^{n-1}x_n \geq \theta_n = \theta_{n-1}$$
4. $f = g \vee \overline{x}_n$: then f is
$$w_1x_1 + \ldots + w_{n-1}x_{n-1} - 2^{n-1}x_n \geq \theta_n = \theta_{n-1} - 2^{n-1}.$$

In each case, it can be verified that the stated threshold function is equivalent to f and that $w_1, \ldots, w_n, \theta_n$ satisfy the conditions of the lemma. ■

3 WINNOW CANNOT LEARN POSITIVE HALFSPACES

Although the Winnow algorithm has been studied extensively in the on-line mistake-bound learning model, little is known about its performance in other learning models. In this section we show that Winnow is not a PAC learning algorithm for the class of positive boolean threshold functions. More precisely, we prove the following theorem:

Theorem 1 *Given any positive start vector w^I, any promotion factor $\alpha > 1$ and any threshold $\theta > 0$, there is a positive threshold function c, a distribution $\mathcal{D}$ on $\{0,1\}^n$, and a value $\epsilon > 0$ for which $Winnow(w^I, \alpha, \theta)$ will not generate a hypothesis h such that $\Pr_{x \in \mathcal{D}}[h(x) \neq c(x)] \leq \epsilon$ in $poly(n, \frac{1}{\epsilon})$ steps.*

As a consequence of the proof of this theorem, we will obtain an explicit family of "hard" threshold functions and corresponding example sequences which cause Winnow to make exponentially many mistakes. Maass and Turan [31] have used a counting argument to show that given any triple (w^I, α, θ), there exists a target threshold function and example sequence which together cause $Winnow(w^I, \alpha, \theta)$ to make exponentially many mistakes, but no explicit construction was known.

3.1 A THRESHOLD FUNCTION WITH LARGE COEFFICIENTS AND A SMALL SPECIFYING SET

The proof of Theorem 1 makes use of several lemmas. In the first lemma, we show that a nested boolean function with alternating connectives requires exponentially large coefficients. (Similar results can be found in [33], [35].)

Lemma 2 *Let n be odd and let $u \cdot x \geq \theta$ be a positive threshold function which represents the nested function*

$$f_n = (\ldots(x_1 \vee x_2) \wedge x_3) \vee x_4) \ldots) \vee x_{n-1}) \wedge x_n.$$

For $3 \leq i \leq n$ we have $u_i \geq F_{i-3}u_3$, where F_i is the i-th Fibonacci number: $F_0 = F_1 = F_2 = 1, F_3 = 2, F_4 = 3$, etc..

Proof: The proof is by induction on k, where $n = 2k + 1$. For clarity we use two base cases. The case $k = 1$ is trivial. If $k = 2$, then since $f_5((0,0,0,1,1)) = 1$ and $f_5((0,0,1,0,1)) = 0$, we find that $u_4 > u_3$. Similarly, since $f_5((0,0,1,1,0)) = 0$, we find that $u_5 > u_3$.

We now suppose that the lemma is true for the values $k = 1, 2, \ldots, j-1$ and let $n = 2j + 1$. By assumption, $(u_1, \ldots, u_n)$ and θ are such that $u \cdot x \geq \theta$ represents f_n. If we fix $x_n = 1$ and $x_{n-1} = 0$, then it follows that $u_1x_1 + \cdots + u_{n-2}x_{n-2} \geq \theta - u_n$ is a threshold function which represents f_{n-2}, so by the induction hypothesis the lemma holds for $u_3, \ldots, u_{n-2}$, and we need only show that it holds for u_{n-1} and u_n.

Since $f_n((1,1,\ldots,1,0,0,1)) = 0$, we have that $u_1 + u_2 + \cdots + u_{n-3} + u_n < \theta$. On the other hand, since $f_n((0,0,\ldots,0,1,1)) = 1$, we have that $u_{n-1} + u_n \geq \theta$. From these two inequalities it follows that

$$u_{n-1} > u_1 + u_2 + \cdots + u_{n-3}.$$

Since u is positive, this inequality implies that

$$u_{n-1} > u_3 + u_4 + \cdots + u_{n-3}.$$

Using the induction hypothesis we obtain the inequality

$$u_{n-1} > (1 + F_1 + \cdots + F_{n-6})u_3 = F_{n-4}u_3.$$

Similarly, since $f_n((1,1,\ldots,1,0)) = 0$, we have that $u_1 + u_2 + \cdots + u_{n-1} < \theta$, so

$$u_n > u_1 + u_2 + \cdots + u_{n-2} > u_3 + u_4 + \cdots + u_{n-2}.$$

By the induction hypothesis, we find that

$$u_n > (1 + F_1 + \cdots + F_{n-5})u_3 = F_{n-3}u_3,$$

and the lemma is proved. ■

Since $F_n = \Omega(\phi^n)$, where $\phi = \frac{1+\sqrt{5}}{2}$, we have shown that f_n must have coefficients whose ratio is exponentially large.

We require a definition from [4] before stating the next lemma.

Definition 3 *Let c be a linear threshold function over $\{0,1\}^n$. We say that c is* consistent *with a subset $S = \{\langle x^1, b_1\rangle, \langle x^2, b_2\rangle, \ldots, \langle x^m, b_m\rangle\}$ of labelled examples if $c(x^i) = b_i$ for all i. The set S is said to* specify *c if c is the only linear threshold function over $\{0,1\}^n$ which is consistent with S; we say that such a set is a* specifying set *for c. The* specification number *of c, denoted $\sigma_n(c)$, is the smallest size of any specifying set for c.*

The following results are proved in [4]:

Lemma 3

1. *Every linear threshold function c over $\{0,1\}^n$ has a unique specifying set of size $\sigma_n(c)$.*
2. *If $c \in NF_n$ then $\sigma_n(c) = n+1$.*
3. *If c is a linear threshold function over $x_1, \ldots, x_n$, let $c\!\uparrow\!((x_1, \ldots, x_{n-1}))$ denote $c((x_1, \ldots, x_{n-1}, 1))$ and let $c\!\downarrow\!((x_1, \ldots, x_{n-1}))$ denote $c((x_1, \ldots, x_{n-1}, 0))$. Then*
$$\sigma_n(c) \leq \sigma_{n-1}(c\!\uparrow) + \sigma_{n-1}(c\!\downarrow).$$
4. *If c is a linear threshold function over $\{0,1\}^n$ which depends only on coordinates $1, 2, \ldots, k$, then the specification number of c is*
$$\sigma_n(c) = 2^{n-k}\sigma_k(c).$$

We use the results of Lemma 3 to show that the function g_n defined below has a small specifying set.

Lemma 4 *Let n be even and let g_n be the boolean threshold function represented by*

$$x_1 + 2x_2 + 4x_3 + \cdots + 2^{n-2}x_{n-1} + (2^{n-2}+1)x_n \geq 4 + 16 + \cdots + 2^{n-4} + 2^{n-2} + \frac{1}{2}.$$

Then $\sigma_n(g_n) \leq 5n - 8$.

Proof: The function $g_n\!\downarrow$ is represented by

$$x_1 + 2x_2 + 4x_3 + \cdots + 2^{n-2}x_{n-1} \geq 4 + 16 + \cdots + 2^{n-2} + \frac{1}{2}.$$

It is straightforward to verify that this is precisely the nested function

$$(\ldots(x_1 \vee x_2) \wedge x_3) \vee x_4) \ldots) \wedge x_{n-1}$$

on $n-1$ variables. By Part 2 of Lemma 3, we have that $\sigma_{n-1}(g_n\!\downarrow) = n$.

The function $g_n\!\uparrow$ is represented by

$$x_1 + 2x_2 + 4x_3 + \cdots + 2^{n-2}x_{n-1} \geq 4 + 16 + \cdots + 2^{n-4} - \frac{1}{2}.$$

Again, one can easily verify that this is precisely the nested function

$$(\ldots(x_3 \vee x_4) \wedge x_5) \vee x_6) \ldots) \wedge x_{n-3}) \vee x_{n-2}) \vee x_{n-1}$$

on the $n-3$ variables $x_3, \ldots, x_{n-1}$; in this nested function the boolean connectives alternate between $\vee$ and $\wedge$ until the very end, where two consecutive $\vee$'s occur. Since $g_n\!\uparrow$ does not depend on the two variables x_1, x_2, parts 4 and 2 of Lemma 3 imply that $\sigma_{n-1}(g_n\!\uparrow) = 4\sigma_{n-3}(g_n\!\uparrow) = 4(n-2)$. It follows from part 3 of Lemma 3 that $\sigma_n(g_n) \leq 5n - 8$. ■

3.2 WINNOW CANNOT PAC LEARN POSITIVE HALFSPACES

One more lemma is required. We prove that the coefficients of x_{n-1} and x_n in any representation of g_n must be almost, but not exactly, equal.

Lemma 5 *Let n be even and let g_n be defined as in Lemma 4. Let $v \cdot x \geq \theta$ represent g_n. Then $v_{n-1} < v_n < v_{n-1} + v_3$.*

Proof: Let e_j denote the boolean vector whose j-th coordinate is 1 and all of whose other coordinates are 0. Let $a = e_3 + e_5 + e_7 + \cdots + e_{n-1}$. Since $g_n(a) = 0$, it follows that $v_3 + v_5 + \cdots + v_{n-1} < \theta$. On the other hand, let $b = e_3 + e_5 + \cdots + e_{n-3} + e_n$. Since $g_n(b) = 1$, it follows that $v_3 + v_5 + \cdots + v_{n-3} + v_n \geq \theta$. Combining these two inequalities, we find that $v_n > v_{n-1}$.

To see that v_n cannot be much greater than v_{n-1}, let $c = e_1 + e_5 + e_7 + e_9 + \cdots + e_{n-3} + e_n$. Since $g_n(c) = 0$, we have $v_1 + v_5 + \cdots + v_{n-3} + v_n < \theta$. On the other hand, let $d = e_1 + e_3 + \cdots + e_{n-1}$. Since $g_n(d) = 1$, we have that $v_1 + v_3 + \cdots + v_{n-1} \geq \theta$. Combining these two inequalities, we find that $v_n < v_{n-1} + v_3$, and the lemma is proved. ■

Proof of Theorem 3: We first prove the theorem for the restricted case in which we assume that $w^I = (1, \ldots, 1)$. After we have done this, we will show how this assumption can be eliminated.

Fix $\alpha > 1$ and $\theta > 0$. Let S denote the specifying set for the threshold function g_n; we know from Lemma 4 that $|S| \leq 5n - 8$. Let $\mathcal{D}$ be the distribution on $\{0,1\}^n$ which is uniform on S and gives zero weight to vectors outside of S. We will show that with g_n as the target concept, $\mathcal{D}$ as the distribution over examples, and $\epsilon = \frac{1}{5n-7}$ as the error parameter, $Winnow((1, \ldots, 1), \alpha, \theta)$ cannot achieve a hypothesis $h(x)$ which satisfies $\Pr_{\mathcal{D}}[h(x) \neq c(x)] \leq \epsilon$ in poly$(n, \frac{1}{\epsilon})$ = poly(n) steps. To see this, note first that by our choice of ϵ and $\mathcal{D}$, any threshold function $w \cdot x \geq \theta$ which is ϵ-accurate with respect to g_n must be consistent with S. (This technique was first used by Pitt and Valiant in [36].) Since S is a specifying set for g_n, though, if $w \cdot x \geq \theta$ is consistent with S then it must in fact agree exactly with g_n. We will show that there is no value of α which could enable Winnow to generate a vector w such that $w \cdot x \geq \theta$ represents $g_n(x)$ in poly(n) steps.

Let (w, θ) be such that Winnow generates w and $w \cdot x \geq \theta$ represents $g_n(x)$. Since $g_n\!\downarrow$ is precisely the nested function f_{n-1} of Lemma 2 and w is positive, by Lemma 2 we have that $w_{n-1} \geq F_{n-4}w_3$. Combining this with Lemma 5, we obtain

$$1 < \frac{w_n}{w_{n-1}} < 1 + \frac{1}{F_{n-4}} = 1 + \frac{1}{\Omega(\phi^n)}.$$

Since we assumed that $w^I = (1, \ldots, 1)$, and every example for Winnow lies in $\{0,1\}^n$, it follows that $\frac{w_n}{w_{n-1}} = \alpha^j$ for some positive integer j, and hence that $\alpha = 1 + \frac{1}{\Omega(\phi^n)}$. But then at least $\Omega(n\phi^n)$ update steps are required to achieve $w_{n-1} \geq F_{n-4}w_3$; consequently, no hypothesis consistent with g_n can be achieved in poly(n) steps.

Now we consider the case of an arbitrary positive start vector w^I; so fix some positive w^I, $\alpha > 1$, and $\theta > 0$. We assume without loss of generality that $w_1^I \leq w_2^I \leq \ldots \leq w_n^I$, since if this is not already the case renaming variables will make it so. Since all examples for Winnow are in $\{0,1\}^n$, at every point in the execution of Winnow the ratio of weights w_i and w_j must be $\frac{w_i^I}{w_j^I} \cdot \alpha^c$ for some integer c. If there is no integer i_1 such that

$$1 < \frac{w_n^I}{w_{n-1}^I} \cdot \alpha^{i_1} < 1 + \frac{1}{F_{n-4}},$$

then Winnow can never achieve a hypothesis which represents the threshold function $g_n(x)$, and thus can never achieve ϵ-accuracy; so we assume that such an i_1 exists. Similarly, if there is no integer i_2 such that

$$1 < \frac{w_{n-1}^I}{w_n^I} \cdot \alpha^{i_2} < 1 + \frac{1}{F_{n-4}},$$

then there is a threshold function which Winnow can never achieve (the function g_n with a permutation on the variables is such a function), so such an i_2 must exist as well. Taking the product of these inequalities, we find that

$$1 < \alpha^{i_1+i_2} < \left(1 + \frac{1}{F_{n-4}}\right)^2.$$

Since $i_1 + i_2$ must be a positive integer, this implies that $\alpha < \left(1+\frac{1}{F_{n-4}}\right)^2$. Now consider a threshold function which requires that $w_1 \geq F_{n-4} w_n$ (again, g_n with a permutation on the variables is such a function). Since $w_1^I \leq w_n^I$ and $\alpha < \left(1+\frac{1}{F_{n-4}}\right)^2$, it follows that $\Omega(n\phi^n)$ update steps will be required; so no consistent hypothesis can be achieved in polynomial time. ■

As an immediate consequence of this proof, we note that the example sequence which simply cycles through S will force Winnow to make exponentially many mistakes on g_n.

4 PERCEPTRON CANNOT LEARN HALFSPACES UNDER UNIFORM DISTRIBUTIONS

In the construction just presented, the "hard" distribution for Winnow depended on the target linear threshold function. In this section we show that a single natural distribution (the uniform distribution over $\{0,1\}^n$) can thwart the Perceptron algorithm. A very simple argument suffices to establish this result. We require the following definition:

Definition 4 *[38] The* weight complexity *of a boolean threshold function f is the smallest natural number t such that f can be represented as $w \cdot x \geq \theta$, with each w_i and θ an integer and max$\{|w_1|, \ldots, |w_n|, |\theta|\} \leq t$.*

Theorem 2 *The Perceptron algorithm is not a PAC learning algorithm for the class of linear threshold functions under the uniform distribution on $\{0,1\}^n$.*

Proof: We take $\epsilon = \frac{1}{2^n+1}$, so any ϵ-accurate hypothesis must agree exactly with the target concept since misclassification of a single example would imply an error rate under the uniform distribution of at least $\frac{1}{2^n} > \epsilon$. Håstad [17] has constructed a boolean threshold function which has weight complexity $2^{\Omega(n \log n)}$. If we take this as our target concept, then it follows that at least $2^{\Omega(n \log n)}$ update steps must be performed by the Perceptron algorithm in order to achieve exact identification (since Perceptron hypothesis weights are always integral and each weight is increased by at most 1 during each Perceptron update step). But the amount of computation time which a PAC learning algorithm is allowed is only poly$(n, \frac{1}{\epsilon})$ = poly$(n, 2^n) = 2^{O(n)}$. ■

5 PERCEPTRON CAN LEARN NESTED FUNCTIONS UNDER UNIFORM DISTRIBUTIONS

In this section we establish a sufficient condition for a class of threshold functions to be efficiently learnable by Perceptron under the uniform distribution on $\{0,1\}^n$. We prove that nested functions satisfy this condition and thereby immediately obtain the main result of this section. This complements Schmitt's result [38] that the Perceptron algorithm cannot efficiently PAC learn nested functions under arbitrary distributions.

Definition 5 *Let G_n be a collection of hyperplanes in R^n. A family $G = \cup_{n \geq 1} G_n$ of hyperplanes is said to be* gradual *if there is some constant $c > 0$ such that the following condition holds: for every $\tau \geq 0$, every $n \geq 1$ and every hyperplane in G_n, at most $c\tau 2^n$ boolean examples $x \in \{0,1\}^n$ lie within distance τ of the hyperplane. A class of boolean threshold functions F is said to be gradual if there is a mapping $\varphi : F \to G$, where G is a gradual family of hyperplanes, such that for all $f \in F$, if $\varphi(f)$ is the hyperplane $w \cdot x = \theta$, then (w, θ) represents the boolean threshold function f.*

Lemma 6 *Nested functions can be represented by gradual boolean threshold functions.*

Proof: We use the representation established in Lemma 1; so let $f \in NF_n$ and let $w \cdot x \geq \theta$ be a linear threshold function which represents f with $w_i = \pm 2^{i-1}$ and $\theta = k + \frac{1}{2}$ for some integer k. For $x \in \{0,1\}^n$, if $w \cdot x = t$ then t must be an integer, but since every integer has a unique binary representation, at most one $x \in \{0,1\}^n$ can satisfy $w \cdot x = t$ for any given value of t. Consequently, no example $x \in \{0,1\}^n$ can have $|w \cdot x - \theta| < \frac{1}{2}$, and

$$|\{x \in \{0,1\}^n : |w \cdot x - \theta| \leq m\}| \leq 2m + 1$$

for $m \geq \frac{1}{2}$. Since the distance from a point x' to the hyperplane $w \cdot x = \theta$ is $\|w\|^{-1} \cdot |w \cdot x' - \theta|$, the lemma follows by noting that we have $\|w\| = (\frac{4^n-1}{3})^{1/2} = \Theta(2^n)$. ■

This lemma ensures that relatively few points can lie close to to the separating hyperplane for a nested function; consequently, as we run Perceptron, most of the updates will cause the algorithm to make significant progress, and it will achieve ϵ-accuracy in polynomial time. The following theorem formalizes this intuition:

Theorem 3 *If C is a gradual class of boolean threshold functions, then the Perceptron is a PAC learning algorithm for C under the uniform distribution on $\{0,1\}^n$.*

Proof: The proof is similar to the proof of Theorem 1 in [7]. Let $w \cdot x \geq \theta$ be a linear threshold function which represents c. We assume without loss of generality that w, θ have been normalized, i.e. $\|w\| = 1$, so $|w \cdot x - \theta|$ is the distance from x to the hyperplane. By Definition 5, there is some constant k such that for all $\tau > 0$, the probability that a random example drawn uniformly from $\{0,1\}^n$ is within distance τ of the hyperplane $w \cdot x = \theta$ is at most $\tau/2k$. Letting $\tau = k\epsilon$, we have that with probability at most $\epsilon/2$, a random example drawn from $\{0,1\}^n$ is within distance $k\epsilon$ of the hyperplane. Let $B \in \{0,1\}^n$ be the set of examples x which lie within distance $k\epsilon$ of the hyperplane, so $\Pr[x \in B] \leq \epsilon/2$.

Let (w_t, θ_t) represent the Perceptron algorithm's hypothesis after t updates have been made. If (w_t, θ_t) is not yet ϵ-accurate, then with probability at most $1/2$ the next example which causes an update will be in B. Consider the following potential function:

$$N_t(\alpha) = \|\alpha w - w_t\|^2 + (\alpha\theta - \theta_t)^2.$$

Recalling our Perceptron update rules, we see that $N_{t+1}(\alpha) - N_t(\alpha)$ is

$$\begin{aligned} \Delta N(\alpha) &= \|\alpha w - w_{t+1}\|^2 + (\alpha\theta - \theta_{t+1})^2 \\ &\quad - \|\alpha w - w_t\|^2 - (\alpha\theta - \theta_t)^2 \\ &= \mp 2\alpha w \cdot x \pm 2\alpha\theta \pm 2w_t \cdot x \mp 2\theta_t + \|x\|^2 + 1 \\ &\leq 2\alpha A \pm 2(w_t \cdot x - \theta_t) + n + 1 \end{aligned}$$

with $A = \mp(w \cdot x - \theta)$. Since x was misclassified, we know that $\pm(w_t \cdot x - \theta_t) < 0$, and hence $\Delta N(\alpha) < 2\alpha A + n + 1$. If $x \in B$, then $A \leq 0$; if $x \notin B$, then $A \leq -k\epsilon$. As a result, $\Delta N(\alpha) < n + 1$ for $x \in B$, and $\Delta N(\alpha) < n + 1 - 2k\epsilon\alpha$ for $x \notin B$. Suppose that during the course of its execution, the Perceptron algorithm has made r updates on examples in B and s updates on examples outside B. Since (w, θ) have been normalized, we have that $|\theta| \leq \sqrt{n}$, and hence $N_0(\alpha) \leq \alpha^2(n+1)$. Since $N_t(\alpha)$ must always be nonnegative, it follows that

$$0 \leq r(n+1) + s(n + 1 - 2k\epsilon\alpha) + \alpha^2(n+1).$$

If we set $\alpha = \frac{12(n+1)}{5k\epsilon}$, then simplifying the above inequality we obtain

$$0 \leq r - \frac{19}{5}s + \frac{144(n+1)^2}{25(k\epsilon)^2},$$

from which it follows that if $m_1 = \frac{144(n+1)^2}{25(k\epsilon)^2}$ updates have been made, at least $7/12$ of the updates must have been on examples in B.

As noted earlier, though, if the Perceptron's hypothesis has never been ϵ-accurate, then at each update step the probability of that update occurring on a point in B is at most $1/2$. By a straightforward application of Chernoff bounds (see [3],[23]) it follows that the probability that more than $7/12$ of $m = \max\{144 \ln \frac{\delta}{2}, m_1\}$ updates occur on points in B is at most $\delta/2$. Consequently, if m updates have been made, then with probability at least $1 - \delta/2$ the Perceptron algorithm will have found an ϵ-accurate hypothesis. Calling $2m/\epsilon$ examples will ensure, with probability at least $1 - \delta/2$, that m updates occur. ■

As an immediate corollary of Theorem 1, we have that the Perceptron is a PAC learning algorithm for the class of nested functions under the uniform distribution on $\{0,1\}^n$.

6 A SIMPLE ALGORITHM FOR ORIGIN-CENTERED HALFSPACES

Up to this point we have been concerned with the broad distinction of whether certain algorithms run in polynomial or superpolynomial time. We now change our focus slightly and give a more detailed running-time analysis of a simple algorithm for learning linear threshold functions with threshold $\theta = 0$ (i.e., origin-centered halfspaces) under the uniform distribution on the unit sphere in R^n.

6.1 PREVIOUS WORK

The problem of learning an unknown origin-centered halfspace in R^n given access to examples drawn uniformly from the unit sphere has been the subject of considerable research [7, 8, 16, 22, 28, 34, 39]. Long [28] proved that any algorithm which learns an origin-centered halfspace to accuracy ϵ under the uniform distribution must use at least $\Omega(\frac{n}{\epsilon})$ examples. Long also showed [29] that by applying Vaidya's linear programming algorithm [40] it is possible to learn to accuracy ϵ in $\tilde{O}(\frac{n^2}{\epsilon} + n^{3.38})$ time steps using $\tilde{O}(\frac{n}{\epsilon})$ examples.[1] (In all of the results which we discuss, as well as in our own results, we adopt the convention that performing a simple arithmetic operation such as multiplication or comparison on a pair of real numbers takes one time step.) Baum [7] showed that the Perceptron algorithm [37] learns to accuracy ϵ using $O(\frac{n}{\epsilon^3})$ examples and running in time $O(\frac{n^2}{\epsilon^3})$. These results of Long and Baum hold in the noise-free model in which it is assumed that the learning algorithm never receives an example which has been labelled incorrectly.

Angluin and Laird [2] introduced the *classification noise* model in which the label of each example is randomly and independently flipped with probability η. Kearns [22] gave a simple algorithm for learning origin-centered halfspaces under the uniform distribution in the presence of classification noise; his algorithm requires $\tilde{O}(\frac{n^2}{\epsilon^2(1-2\eta)^2})$ examples and runs in time $\tilde{O}(\frac{n^3}{\epsilon^2(1-2\eta)^2})$. Blum et al. [9] and Cohen [13] have recently given polynomial-time algorithms for learning halfspaces in the presence of classification noise under an arbitrary distribution. Their algorithms have time complexity at least $O(\frac{n^{14}}{\epsilon^2(1-2\eta)^2})$, though, and hence are not particularly efficient for the uniform distribution problem.

An even more challenging noise model was proposed by Bylander [11], who introduced the notion of *monotonic noise* for halfspaces. In this model the probability that an example is labelled incorrectly decreases with the distance of the example from the separating hyperplane; this is a natural way

[1] We use the notation $\tilde{\Theta}(f)$ to mean "$\Theta(f \cdot p(\log f, \log n, \log \frac{1}{\epsilon}, \log \frac{1}{\delta}, \log \frac{1}{1-2\eta}))$ for some polynomial $p(\cdot,\cdot,\cdot,\cdot,\cdot)$," i.e. $\tilde{\Theta}(f)$ is $\Theta(f)$ up to polylogarithmic factors in these five parameters. We use $\tilde{O}(f)$ analogously.

to model the idea that examples closer to the decision boundary are harder to label correctly. Bylander gave a variant of the Perceptron algorithm which, for certain distributions, learns an unknown halfspace even in the presence of monotonic noise. Under the uniform distribution his monotonic noise algorithm requires $\tilde{O}(\frac{n^3}{\epsilon^4(1-2\eta)^4})$ examples and runs in time $\tilde{O}(\frac{n^5}{\epsilon^6(1-2\eta)^6})$.

6.2 OUR RESULTS

We describe a simple algorithm which learns origin-centered halfspaces under the uniform distribution on the unit sphere. In the presence of monotonic or classification noise our algorithm requires $\tilde{O}(\frac{n}{\epsilon^2(1-2\eta)^2})$ examples and runs in time $\tilde{O}(\frac{n^2}{\epsilon^2(1-2\eta)^2})$. Our algorithm improves on the time and sample complexity of previous algorithms by a polynomial factor in the classification noise and monotonic noise models. The algorithm works by simply summing a large set of labelled examples to obtain a hypothesis vector; it thus learns using the same hypothesis class (linear threshold functions) as the earlier algorithms mentioned above.

6.3 PRELIMINARIES

In this subsection we provide some definitions, describe the noise models which we will consider, and give some useful probability facts.

We denote the n-dimensional unit sphere by

$$S^{n-1} \equiv \{x \in R^n : \|x\| = 1\}$$

(note that S^{n-1} lies in R^n and not in R^{n-1}). An *origin-centered halfspace* over S^{n-1} is a function from S^{n-1} to $\{-1, 1\}$ which is defined by a nonzero vector $u \in R^n$; the value of $u(x)$ is 1 if $u \cdot x \geq 0$ and is -1 if $u \cdot x < 0$.

In the noise-free scenario which we have considered thus far in the paper, the learning algorithm will never encounter an example which has been labelled incorrectly with $-u(x)$ instead of $u(x)$. Bylander [11] has proposed a demanding noise model for learning linear threshold functions in which examples closer to the decision boundary are more likely to be mislabelled. In this model the learner cannot access the oracle EX, but instead must use the noisy example oracle EX^{η}_{MN} where $0 \leq \eta < 1/2$ is a fixed noise rate. Given a halfspace defined by a unit vector $u \in S^{n-1}$, the monotonic noise oracle $EX^{\eta}_{MN}(u, \mathcal{D})$ is defined as follows: the oracle first draws an example x according to $\mathcal{D}$ and then flips a biased coin with $\Pr[\text{heads}] = \hat{\eta}(|u \cdot x|)$. The oracle outputs $\langle x, -u(x)\rangle$ if the coin comes up heads and outputs $\langle x, u(x)\rangle$ if it comes up tails. The function $\hat{\eta} : [0, 1] \to [0, 1]$ can be any nonincreasing function such that the overall probability that the oracle returns a mislabelled example is η. (Note that the classification noise model is obtained when $\hat{\eta}(\cdot)$ is restricted to be the constant function η, and the original noise-free model is obtained when $\eta = 0$.) The learner's goal still is to output a hypothesis h which, with probability at least $1 - \delta$, has $\Pr_{x\in\mathcal{D}}[h(x) \neq u(x)] \leq \epsilon$, but now the learner is given access to $EX^{\eta}_{MN}(u, \mathcal{D})$, ϵ, δ, and η, and the learner is allowed time polynomial in $n, \frac{1}{\epsilon}, \frac{1}{\delta}$, and $\frac{1}{1-2\eta}$. The dependence on $\frac{1}{1-2\eta}$ is necessary because there is less and less information in the labelling of each example as η approaches $1/2$.

Now we give some useful probability facts. We let $\mathcal{U}_{n-1}$ denote the uniform distribution on S^{n-1}. If $u \in S^{n-1}$ represents the target halfspace and $v \in S^{n-1}$ represents a hypothesis halfspace, then $\Pr_{x\in\mathcal{U}_{n-1}}[v(x) \neq u(x)]$, the error of v with respect to u under $\mathcal{U}_{n-1}$, is the fraction of S^{n-1} which lies in the symmetric difference of the two halfspaces. This is easily seen to be $\theta(u, v)/\pi$, where $\theta(u, v) = \arccos(u \cdot v)$ is the angle between u and v.

Let A_{n-1} denote the surface area of the unit sphere S^{n-1}. It is known (see, e.g., [7]) that $A_{n-1} = 2\pi^{n/2}/\Gamma(n/2)$, where Γ is the classical gamma function. Using basic properties of the gamma function [5] one can then show that $A_{n-2}/A_{n-1} = \Theta(n^{1/2})$ (this is our only use of the gamma function in this paper). For $n \geq 3$, if $u \in S^{n-1}$ is fixed, then as noted in [7] we have

$$\Pr_{x\in\mathcal{U}_{n-1}}[\alpha \leq u \cdot x \leq \beta] = \frac{A_{n-2}}{A_{n-1}} \cdot \int_{\alpha}^{\beta} \left(\sqrt{1 - z^2}\right)^{n-3} dz.$$

Finally, we will also use the following tail bound:

Fact 1 (Hoeffding [20]) *Let $X_1, \ldots, X_t$ be independent random variables each of which lies in the range $[a, b]$. If $X = \frac{1}{t}\sum_{i=1}^{t} X_i$, then for $\nu > 0$ we have*

$$\Pr[|E[X] - X| \geq \nu] \leq 2 \cdot \exp\left(\frac{-2t\nu^2}{(b - a)^2}\right).$$

6.4 THE ALGORITHM

Throughout the rest of the paper u denotes the unit vector which represents the target halfspace and MX denotes the monotonic example oracle $EX^{\eta}_{MN}(u, \mathcal{U}_{n-1})$. The learning algorithm we consider, called AVERAGE, is given below.

AVERAGE(MX, t):
1. **for** $i = 1$ **to** t **do**
2. Draw $\langle x^i, b_i\rangle$ from MX
3. **return** $v = \frac{1}{t}\sum_{i=1}^{t} b_i x^i$

It is trivial to verify that AVERAGE(MX, t) uses t examples and runs in time $\Theta(nt)$.

6.4.1 Comparison of AVERAGE and Perceptron

It is interesting to note that the AVERAGE algorithm is very similar to the Perceptron algorithm. The only difference between the two algorithms, in fact, is that Perceptron is *conservative*, i.e., it only updates its current hypothesis v on an example if v predicts incorrectly on that example. The AVERAGE algorithm, on the other hand, can be viewed as performing an update on every example. Baum [7] has shown that in the absence of noise the Perceptron algorithm achieves an ϵ-accurate hypothesis with high probability after $O(n/\epsilon^2)$ updates. Once the Perceptron hypothesis has achieved $\Theta(\epsilon)$ accuracy, though, it requires an expected $\Theta(1/\epsilon)$ examples to generate a single update vector, and hence $O(n/\epsilon^3)$ examples are required overall. We prove in Section 6.5 that in the absence of noise the AVERAGE algorithm requires $\tilde{O}(n/\epsilon^2)$ updates. However, since AVERAGE performs an update on every example, it needs only $\tilde{O}(n/\epsilon^2)$ examples in total. This difference between the two algorithms appears to be a real one and not just an artifact of the analysis. Figure

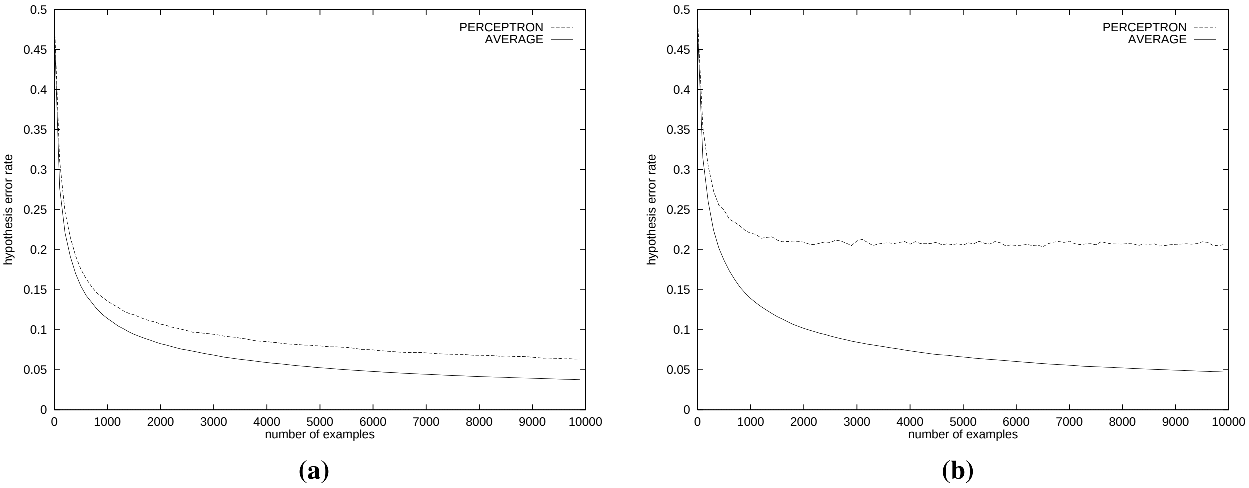

Figure 1: Part (a) shows the error rates of AVERAGE and Perceptron on noise-free examples drawn from $\mathcal{U}_{n-1}$ with $n = 100$. Part (b) shows the error rates of AVERAGE and Perceptron on examples drawn from $\mathcal{U}_{n-1}$ with $n = 100$ and a classification noise rate of $\eta = 0.10$.

1(a) graphs the error rates of the two algorithms as a function of the number of examples used. (The graphs shown in Figures 1(a) and 1(b) are averages over 100 independent runs of each algorithm.)

Another difference between the two algorithms is their ability to tolerate noise. We prove in Section 6.5 that AVERAGE can achieve an arbitrarily accurate hypothesis even in the presence of monotonic noise. No such guarantees are known for Perceptron, and it seems likely that none can be given: Figure 1(b) graphs the error rates of the two algorithms when given examples with a classification noise rate of $\eta = 0.10$. One possible explanation for Perceptron's inability to achieve a low error rate is the following: Initially Perceptron's error rate decreases (see Figure 1(b)). Once the error rate dips below a certain level, though, the algorithm performs most of its updates on mislabelled examples. Performing these "bad" updates then causes the hypothesis to become less accurate, though, so the error rate increases again. In contrast, since AVERAGE updates on every example (and most examples are not mislabelled, since $\eta < 1/2$), its error rate continues to decrease.

6.4.2 Why AVERAGE Works

The idea underlying the AVERAGE algorithm is very simple. If there were no noise in the labelling of the examples, then for each labelled example $\langle x, b\rangle$ it would be the case that $u \cdot (bx) \geq 0$. Furthermore, since the distribution of x is uniform on the unit sphere, the distribution of bx would be symmetrical around the vector u, and hence the expected value of bx would be a vector which points precisely in the direction of u (see Figure 2). In fact, even in the presence of monotonic noise at a noise rate of $\eta < 1/2$, it is still the case that $E[bx]$ is a vector which points precisely in the direction of u (recall that for an example x, the probability that x is corrupted by monotonic noise depends only on $|u \cdot x|$). Thus, if we can approximate $E[bx]$, we can find an accurate hypothesis; the AVERAGE algorithm simply uses sampling to approximate the expected value of bx.

The analysis of the algorithm, therefore, consists of proving that for a suitable value of t, the angle between u and v (and hence the error of the hypothesis v) does not deviate very much from zero. We will do this in two phases: in Section 6.5.1 we show that with high probability the vector v will have a large component lying in the direction of u, and in Section 6.5.2 we prove that with high probability v will have only a small component which is orthogonal to u. Finally, in Section 6.6, we combine these results to complete the proof.

6.5 ANALYZING THE AVERAGE ALGORITHM

6.5.1 A Large Parallel Component

Our first lemma is a lower bound on the expectation of $u \cdot (bx)$. We will use this bound later to prove the main result of this subsection.

Lemma 7 *Let $\langle x, b\rangle$ be obtained by querying MX. Then we have that*

$$E[u \cdot (bx)] = \Omega\left(\frac{1-2\eta}{n^{1/2}}\right).$$

Proof: Let $\hat{\eta}$ be the monotonic noise function; we first show that $E[u \cdot (bx)] = \Omega(\frac{1-2\eta}{n^{1/2}})$ if $\hat{\eta}$ is the constant function η. If this is the case then for each $x \in S^{n-1}$ we have that $\Pr[b = u(x)] = 1 - \eta$ and $\Pr[b \neq u(x)] = \eta$, so

$$\begin{aligned} E[u \cdot (bx)] &= 2 \cdot \frac{A_{n-2}}{A_{n-1}} \cdot \int_0^1 (1-2\eta) z \left(\sqrt{1-z^2}\right)^{n-3} dz \\ &= \Omega\left((1-2\eta) n^{1/2} \int_0^{n^{-1/2}} z \left(\sqrt{1-z^2}\right)^{n-3} dz\right) \\ &= \Omega\left((1-2\eta) n^{1/2} \int_0^{n^{-1/2}} z\, dz\right) \end{aligned}$$

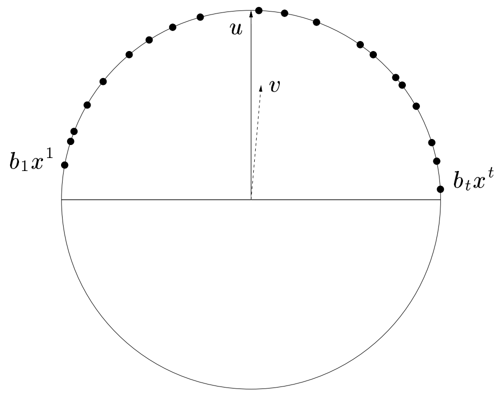

Figure 2: In the noise-free case each point $b_i x^i$ lies above the target hyperplane defined by the vector u. Here v is the vector average of $b_1x^1, \ldots, b_tx^t$.

$$= \Omega\left(\frac{1-2\eta}{n^{1/2}}\right).$$

The second equality holds because the integrand is never negative, and the third equality holds since $(\sqrt{1-z^2})^{n-3} = \Theta(1)$ for $0 \leq z \leq n^{-1/2}$.

We now show that $E[u \cdot (bx)]$ is minimized when $\hat{\eta}(\cdot)$ is the constant function η. Since

$$E[u\cdot(bx)] = 2\cdot\frac{A_{n-2}}{A_{n-1}}\cdot\int_0^1 z\cdot(1-2\hat{\eta}(z))\left(\sqrt{1-z^2}\right)^{n-3}dz$$

it suffices to prove that

$$\int_0^1 \hat{\eta}(z)z\cdot\left(\sqrt{1-z^2}\right)^{n-3}dz \leq \int_0^1 \eta z\cdot\left(\sqrt{1-z^2}\right)^{n-3}dz$$

for any function $\hat{\eta}$ which satisfies the conditions of monotonic noise. Since the overall probability that $b \neq u(x)$ is η, we have

$$\begin{aligned}\eta &= 2\eta\cdot\frac{A_{n-2}}{A_{n-1}}\cdot\int_0^1\left(\sqrt{1-z^2}\right)^{n-3}dz \\ &= 2\cdot\frac{A_{n-2}}{A_{n-1}}\cdot\int_0^1\hat{\eta}(z)\left(\sqrt{1-z^2}\right)^{n-3}dz.\end{aligned}$$

This implies that for all $0 \leq \beta \leq 1$,

$$\begin{aligned}&\int_0^\beta(\hat{\eta}(z)-\eta)\left(\sqrt{1-z^2}\right)^{n-3}dz \\ &\quad = \int_\beta^1(\eta-\hat{\eta}(z))\left(\sqrt{1-z^2}\right)^{n-3}dz.\end{aligned}$$

Now let us choose β such that $\hat{\eta}(z) \geq \eta$ for $0 \leq z < \beta$ and $\hat{\eta}(z) \leq \eta$ for $\beta < z \leq 1$ (such a β must exist because $\hat{\eta}$ is a nonincreasing function and the overall probability of misclassification is η). We then have

$$\begin{aligned}&\int_0^\beta z\cdot(\hat{\eta}(z)-\eta)\left(\sqrt{1-z^2}\right)^{n-3}dz \\ \leq\ & \int_0^\beta \beta\cdot(\hat{\eta}(z)-\eta)\left(\sqrt{1-z^2}\right)^{n-3}dz \\ =\ & \int_\beta^1 \beta\cdot(\eta-\hat{\eta}(z))\left(\sqrt{1-z^2}\right)^{n-3}dz \\ \leq\ & \int_\beta^1 z\cdot(\eta-\hat{\eta}(z))\left(\sqrt{1-z^2}\right)^{n-3}dz.\end{aligned}$$

This implies the desired inequality, so we have shown that $E[u\cdot(bx)] = \Omega(\frac{1-2\eta}{n^{1/2}})$ for any monotonic noise function $\hat{\eta}$. ■

Now we can prove the main result of this section, which establishes that the output of AVERAGE(MX, t) will, with high probability, have a substantial component in the direction of u.

Lemma 8 *Let v = AVERAGE(MX,t). Then there is some value $t = \tilde{\Theta}(\frac{n}{(1-2\eta)^2})$ such that*

$$u\cdot v = \Omega\left(\frac{1-2\eta}{n^{1/2}}\right)$$

with probability at least $1-\delta/2$.

Proof: Since $u \cdot v$ is $\frac{1}{t}$ times a sum of t independent random variables, we can use Fact 1 to obtain

$$\Pr\left[u\cdot v \leq \frac{1}{2}E[u\cdot v]\right] \leq 2\cdot\exp\left(\frac{-t(E[u\cdot v])^2}{8}\right).$$

By linearity of expectation and Lemma 7 we have that

$$E[u\cdot v] = \Omega\left(\frac{1-2\eta}{n^{1/2}}\right).$$

Hence in order to bound the above probability by $\delta/2$, it suffices to take

$$t = \Theta\left(\frac{n\log\frac{1}{\delta}}{(1-2\eta)^2}\right) = \tilde{\Theta}\left(\frac{n}{(1-2\eta)^2}\right).$$

■

6.5.2 A Small Orthogonal Component

Now we must show that with high probability the component of v which is orthogonal to u is not very large. The following lemma shows that with high probability a random flight in R^n which proceeds for t steps will end up at most a distance of (approximately) $t^{1/2}$ away from its starting point. We will use this lemma to prove our desired upper bound.

Lemma 9 *Let $x^1,\ldots,x^t$ be independently drawn from $\mathcal{U}_{n-1}$. Then with probability at least $1-\delta/2$ we have*

$$\left\|\sum_{i=1}^t x^i\right\| = \tilde{O}(t^{1/2}).$$

Proof: Consider a fixed coordinate $j \in \{1,\ldots,n\}$. Since $\sum_{i=1}^t x_j^i$ is a sum of independent random variables each of which lies in the range $[-1,1]$, we could simply apply Fact 1 to obtain a bound on the probability that $\sum_{i=1}^t x_j^i$ deviates significantly from its expected value of zero. This straightforward approach, however, yields a weaker final bound than one which we can obtain with a little additional work by using conditional probabilities and expectations. By conditioning on the magnitude of each x_j^i, we can replace $[-1,1]$ with a much smaller interval and thereby obtain a tighter bound.

For any fixed coordinate $j \in \{1,\ldots,n\}$ and any $\alpha \geq 0$, we have that

$$\begin{aligned}\Pr_{x\in\mathcal{U}_{n-1}}[|x_j| \geq \alpha] &= 2\cdot\frac{A_{n-2}}{A_{n-1}}\cdot\int_\alpha^1\left(\sqrt{1-z^2}\right)^{n-3}dz \\ &\leq 2\cdot\frac{A_{n-2}}{A_{n-1}}\cdot\left(\sqrt{1-\alpha^2}\right)^{n-3}.\end{aligned}$$

Consequently for $\alpha = \omega(\log^{1/2}(tn/\delta)/n^{1/2})$, we have

$$\Pr[|x_j| \geq \alpha] < \frac{\delta}{4tn}.$$

Let $\mathcal{C}$ denote the event that $|x_j^i| < \alpha$ for $1 \leq i \leq t$ (so $\Pr[\mathcal{C}] \geq 1-\delta/4n$ by the union bound). Since by symmetry we have $E[\sum_{i=1}^t x_j^i \mid \mathcal{C}] = 0$, we can apply Fact 1 to obtain the following: for any $\nu \geq 0$,

$$\Pr\left[\left|\frac{1}{t}\sum_{i=1}^t x_j^i\right| > \nu \mid \mathcal{C}\right] \leq 2\cdot\exp\left(\frac{-t\nu^2}{2\alpha^2}\right).$$

Taking $\alpha = \tilde{\Theta}(n^{-1/2})$ and $\nu = \tilde{O}((nt)^{-1/2})$, this probability can be bounded by $\delta/4n$. Since $\Pr[\mathcal{C}] \geq 1-\delta/4n$, this implies that with probability at least $1-\delta/2n$ we have

$$\left(\sum_{i=1}^t x_j^i\right)^2 = \tilde{O}\left(\frac{t}{n}\right).$$

Using the union bound over $x_1,\ldots,x_n$, we find that with probability at least $1-\delta/2$ we have

$$\left(\sum_{i=1}^t x_1^i\right)^2 + \cdots + \left(\sum_{i=1}^t x_n^i\right)^2 = \tilde{O}(t)$$

which proves the lemma. ■

Now we use Lemma 9 to give an upper bound on the magnitude of the component of v which is orthogonal to u.

Lemma 10 *Let $v = \text{AVERAGE}(MX,t)$ and let v' be the component of v which is orthogonal to u (i.e. $v' = v - (u\cdot v)u$). Then with probability at least $1-\delta/2$ we have that*

$$\|tv'\|^2 = \tilde{O}(t).$$

Proof: Without loss of generality we can suppose that the target vector u is $(1,0,\ldots,0)$, so our goal is to bound

$$t^2(v_2^2+\cdots+v_n^2).$$

Recall that $tv = \sum_{i=1}^t b_i x^i$. Since each x^i is drawn from $\mathcal{U}_{n-1}$, the distribution of each x_j^i is symmetric around 0. For any $i \in \{1,\ldots,t\}$ and $j \in \{2,\ldots,n\}$ the distribution of b_i is independent of the distribution of x_j^i, so we have that the distribution of $b_i x_j^i$ is identical to that of x_j^i, and hence the distribution of $\sum_{i=1}^t b_i x_j^i$ is identical to that of $\sum_{i=1}^t x_j^i$ for $j = 2,\ldots,n$. Since

$$t^2 v_j^2 = \left(\sum_{i=1}^t b_i x_j^i\right)^2$$

for $j = 2,\ldots,n$, the distribution of $t^2(v_2^2+\cdots+v_n^2)$ is identical to that of

$$\left(\sum_{i=1}^t x_2^i\right)^2 + \cdots + \left(\sum_{i=1}^t x_n^i\right)^2.$$

Lemma 9 implies that

$$\left(\sum_{i=1}^t x_2^i\right)^2 + \cdots + \left(\sum_{i=1}^t x_n^i\right)^2 = \tilde{O}(t)$$

with probability at least $1-\delta/2$, so the lemma is proved. ■

6.6 PUTTING IT ALL TOGETHER

We have almost reached our goal. By Lemma 8, if we take t sufficiently large, then with probability at least $1-\delta/2$ we have

$$u\cdot v = \Omega\left(\frac{1-2\eta}{n^{1/2}}\right).$$

On the other hand, Lemma 10 tells us that

$$\|tv'\| = \tilde{O}(t^{1/2})$$

with probability at least $1-\delta/2$. Consequently, with probability $1-\delta$ we have

$$\frac{\|v'\|}{u\cdot v} = \tilde{O}\left(\frac{n^{1/2}}{t^{1/2}(1-2\eta)}\right).$$

Note that $\|v'\|/(u\cdot v)$ is the tangent of $\theta(u,v)$, the angle between u and v. Also, as mentioned in Section 6, the error of v under $\mathcal{U}_{n-1}$, which we denote by $\mathit{error}(v)$, is $\theta(u,v)/\pi$. Consequently, with probability $1-\delta$ we have

$$\begin{aligned}\mathit{error}(v) &= \frac{1}{\pi}\cdot\arctan\left(\frac{\|v'\|}{u\cdot v}\right) \\ &\leq \frac{\|v'\|}{(u\cdot v)\cdot\pi} \\ &= \tilde{O}\left(\frac{n^{1/2}}{t^{1/2}(1-2\eta)}\right).\end{aligned}$$

(the inequality holds because $\arctan(x) \leq x$ for $x \geq 0$). By choosing $t = \tilde{\Theta}(\frac{n}{\epsilon^2(1-2\eta)^2})$ we can satisfy all of the required conditions and obtain $\mathit{error}(v) < \epsilon$. Thus, we have proved the following theorem:

Theorem 4 *For some value of $t = \tilde{\Theta}(\frac{n}{\epsilon^2(1-2\eta)^2})$, the algorithm $\text{AVERAGE}(MX,t)$ is a PAC learning algorithm for the class of origin-centered halfspaces under $\mathcal{U}_{n-1}$ which uses $\tilde{\Theta}(\frac{n}{\epsilon^2(1-2\eta)^2})$ examples and runs in $\tilde{\Theta}(\frac{n^2}{\epsilon^2(1-2\eta)^2})$ time steps.*

7 CONCLUSION

Many questions remain to be answered about the PAC learning ability of simple algorithms such as Perceptron and Winnow. Perceptron is now known to be an efficient PAC learning algorithm for linear threshold functions under the uniform distribution on S^n and is known not to be an efficient PAC learning algorithm for linear threshold functions under the uniform distribution on $\{0, 1\}^n$. Analogous results have yet to be obtained for Winnow under uniform distributions. More generally, it would be interesting to identify the class of threshold functions which Perceptron and Winnow can PAC learn under the uniform distribution and under arbitrary distributions.

Another interesting issue is attribute efficiency. A PAC algorithm is said to be *attribute efficient* if, when the target concept has k relevant variables out of n, the number of examples which the learning algorithm uses is polynomial in k and logarithmic in n. Winnow is known to be an attribute efficient algorithm for certain simple subclasses of boolean threshold functions such as disjunctions and r-out-of-k threshold functions [25]. The results of this paper show that Winnow is not an attribute-efficient learning algorithm for the unrestricted class of boolean threshold functions. It would be interesting to gain a better understanding of the abilities and limitations of Winnow as an attribute efficient learning algorithm.

8 ACKNOWLEDGEMENTS

I would like to thank Amos Beimel, Michael Mitzenmacher, and Les Valiant for their very helpful feedback on a preliminary version of this paper.

References

[1] D. Angluin. Queries and concept learning. *Machine Learning,* 2(4):319-342, 1988.

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

[3] D. Angluin and L. G. Valiant. Fast probabilistic algorithms for Hamiltonian circuits and matchings. *J. of Computer and System Sciences,* 18(2):155-193, 1979.

[4] M. Anthony, G. Brightwell, and J. Shawe-Taylor. On specifying boolean functions using labelled examples. *Discrete Applied Math.,* 61(1):1-25, 1993.

[5] E. Artin. The Gamma Function. Holt, Rinehart and Winston, New York, 1964.

[6] P. Auer and M. Warmuth. Tracking the best disjunction. In "36th Symposium on Foundations of Computer Science" pages 312-321, 1995.

[7] E. B. Baum. The perceptron algorithm is fast for nonmalicious distributions. *Neural Computation,* 2:248-260 ,1990.

[8] E. B. Baum and Y-D. Lyuu. The transition to perfect generalization in perceptrons. *Neural Computation,* 3:386-401, 1991.

[9] A. Blum, A. Frieze, R. Kannan, and S. Vempala. A polynomial time algorithm for learning noisy linear threshold functions. In "37th Symposium on Foundations of Computer Science," pages 330-338,1996.

[10] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *J. ACM,* 36(4):929-965, 1989.

[11] T. Bylander. Learning noisy linear threshold functions. Submitted for publication, available at `http://ringer.cs.utsa.edu/~bylander/pubs/pubs.html`, 1998.

[12] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Ann. Math. Statist.,* 23:493-507, 1952.

[13] E. Cohen. Learning noisy perceptrons by a perceptron in polynomial time. In "38th Symposium on Foundations of Computer Science,", pages 514-523, 1997.

[14] M. L. Dertouzos. Threshold Logic: A Synthesis Approach. MIT Press, Cambridge, MA, 1965.

[15] Y. Freund and R. Schapire. Large margin classification using the Perceptron algorithm. In "11th Conference on Computational Learning Theory," pages 209-217, 1998.

[16] E. Gardner and B. Derrida. Three unfinished works on the optimal storage capacity of networks. *J. Phys. A: Math. Gen.,* 22:1983-1994, 1989.

[17] J. Håstad. On the size of weights for threshold gates. *SIAM J. Discrete Math.,* 7(3):484-492, 1994.

[18] S. Hampson and D. Volper. Linear function neurons: structure and training. *Biological Cybernetics,* 53:203-217, 1986.

[19] D. Haussler. Space efficient learning algorithms. Technical Report UCSC-CRL-88-2, University of Calif., Santa Cruz, 1988.

[20] W. Hoeffding. Probability inequalities for sums of bounded random variables. *J. Amer. Statist. Assoc.,* 58:13-30, 1963.

[21] N. Karmarkar. A new polynomial time algorithm for linear programming. *Combinatorica,* 4:373-395, 1984.

[22] M. Kearns. Efficient Noise-Tolerant Learning from Statistical Queries. In "25th Symposium on Theory of Computation," pages 392-401, 1993.

[23] M. Kearns and U. Vazirani. An Introduction to Computational Learning Theory. MIT Press, Cambridge, MA, 1994.

[24] J. Kivinen, M. Warmuth, and P. Auer. The perceptron algorithm vs. winnow: linear vs. logarithmic mistake bounds when few input variables are relevant. In "8th Conference on Computational Learning Theory," pages 289-296, 1995.

[25] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning,* 2:285-318, 1988.

[26] N. Littlestone. Mistake Bounds and Logarithmic Linear-Threshold Learning Algorithms. Ph.D. thesis, Technical Report UCSC-CRL-89-11, University of Calif., Santa Cruz, 1989.

[27] N. Littlestone. Redundant noisy attributes, attribute errors, and linear-threshold learning using winnow. In "4th Conference on Computational Learning Theory," pages 147-156, 1991.

[28] P. Long. On the sample complexity of PAC learning halfspaces against the uniform distribution. In *IEEE Trans. on Neural Networks,* 6(6):1556-1559, 1995.

[29] P. Long. Halfspace learning, linear programming, and nonmalicious distributions. *Information Processing Letters,* 51:245-250, 1994.

[30] W. Maass and M. Warmuth. Efficient learning with virtual threshold gates. Technical Report UCSC-CRL-95-37, University of Calif., Santa Cruz, 1995.

[31] W. Maass and G. Turan. How fast can a threshold gate learn? in "Computational Learning Theory and Natural Learning Systems: Volume I: Constraints and Prospects," S. J. Hanson, G. Drastal, & R. Rivest, eds., MIT Press, Cambridge, MA, pages 381-414, 1994.

[32] M. Minsky and S. Papert. Perceptrons: An Introduction to Computational Geometry. Expanded edition, MIT Press, Cambridge, MA, 1988.

[33] S. Muroga. Threshold Logic and its Applications. Wiley-Interscience, New York, 1971.

[34] M. Opper and D. Haussler. Calculation of the learning curve of Bayes optimal classification algorithm for learning a perceptron with noise. In "4th Conference on Computational Learning Theory," pages 75-87, 1991.

[35] I. Parberry. Circuit Complexity and Neural Networks. MIT Press, Cambridge, MA, 1994.

[36] L. Pitt and L. G. Valiant. Computational limitations on learning from examples. *J. ACM,* 35(4):965-984, 1988.

[37] F. Rosenblatt. Principles of Neurodynamics. Springer-Verlag, New York, 1962.

[38] M. Schmitt. Identification criteria and lower bounds for Perceptron-like learning rules. *Neural Computation,* 10:235-250, 1998.

[39] H. S. Seung, H. Sompolinsky, and N. Tishby. Statistical mechanics of learning from examples. *Physical Review A,* 45(8):6056-6091, 1992.

[40] P. Vaidya. A new algorithm for minimizing convex functions over convex sets. In "30th Symposium on Foundations of Computer Science," pages 338-343, 1989.

[41] L. G. Valiant. A theory of the learnable. *Comm. ACM,* 27(11):1134-1142, 1984.

[42] L. G. Valiant. Projection learning. In "11th Conference on Computational Learning Theory," pages 287-293, 1998.