SIAM J. Comput.
Vol. 31, No. 5, pp. 1358–1369

PERCEPTRON, WINNOW, AND PAC LEARNING[*]

ROCCO A. SERVEDIO[†]

Abstract. We analyze the performance of the widely studied Perceptron and Winnow algorithms for learning linear threshold functions under Valiant's probably approximately correct (PAC) model of concept learning. We show that under the uniform distribution on boolean examples, the Perceptron algorithm can efficiently PAC learn nested functions (a class of linear threshold functions known to be hard for Perceptron under arbitrary distributions) but cannot efficiently PAC learn arbitrary linear threshold functions. We also prove that Littlestone's Winnow algorithm is not an efficient PAC learning algorithm for the class of positive linear threshold functions, thus answering an open question posed by Schmitt [*Neural Comput.*, 10 (1998), pp. 235–250]. Based on our results we conjecture that no "local" algorithm can learn linear threshold functions efficiently.

Key words. linear threshold function, PAC learning, Perceptron, Winnow

AMS subject classification. 68Q32

PII. S0097539798340928

1. Introduction.

The classical Perceptron algorithm and Littlestone's Winnow algorithm are two algorithms for learning linear threshold functions which have been studied extensively in the online mistake bound model [4, 8, 13, 16, 17, 19, 28]. 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 prediction mistakes it makes on any example sequence. The well-known Perceptron convergence theorem [6, 20, 24] establishes conditions under which the Perceptron algorithm will make a bounded number of mistakes on a (possibly infinite) sequence of examples, and Littlestone [14, 15, 16] has obtained similar results for the Winnow algorithm.

Despite widespread interest in the Perceptron and Winnow algorithms, relatively little is known about their performance in Valiant's commonly used probably approximately correct (PAC) model of concept learning [26]. In this paper we establish some positive and negative results which help to clarify the learning abilities of Perceptron and Winnow in the PAC model.

In the PAC learning 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, 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 close to the separating hyperplane. Baum [5] has shown that if $\mathcal{D}$ is restricted to be the uniform distribution on S^n, though, then Perceptron is an efficient PAC learning algorithm for the class of linear threshold functions. Schmitt [25] has shown that Perceptron is not an efficient PAC learning algorithm for the class of linear threshold functions over the example space $\{0, 1\}^n$; his proof works by exhibiting a

[*]Received by the editors June 19, 1998; accepted for publication (in revised form) January 23, 2002; published electronically May 29, 2002. This work was supported by an NSF Graduate Fellowship and by grants NSF-CCR-95-04436 and ONR-N00014-96-1-0550.

http://www.siam.org/journals/sicomp/31-5/34092.html

[†]Maxwell Dworkin 147, Division of Engineering and Applied Sciences, Harvard University, Cambridge MA 02138 (rocco@deas.harvard.edu).

nested boolean function (a special type of linear threshold 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 noted, it was not known "whether Littlestone's rules can PAC identify in polynomial time" [25].

This paper makes the following contributions: In section 3 we show that under the uniform distribution on $\{0,1\}^n$ the Perceptron algorithm is an efficient PAC learning algorithm for the class of nested boolean functions. This demonstrates that Schmitt's negative result for nested functions depends on choosing a "hard" distribution. However, in section 4 we give a simple proof that the Perceptron algorithm is not an efficient PAC learning algorithm for the class of linear threshold functions under the uniform distribution on $\{0,1\}^n$. This provides an interesting contrast to Baum's result and answers an open question in Schmitt [25]. Finally, we prove in section 5 that the Winnow algorithm is not an efficient PAC learning algorithm for the class of positive linear threshold functions over $\{0,1\}^n$. To the best of our knowledge this is the first negative result for Winnow in the PAC model. We conclude in section 6 with suggestions for future research.

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 $X = \cup_{n\geq 1} X_n$, where each $c \in C_n$ has domain X_n. Throughout this paper X_n will be the boolean cube $\{0,1\}^n$. We will mainly deal with the class H_n of n-dimensional linear threshold functions. A boolean function $f : \{0,1\}^n \to \{-1,1\}$ is a linear threshold function if there is a weight vector $w \in \mathbb{R}^n$ and a threshold $\theta \in \mathbb{R}$ such that $f(x) = 1$ iff $w \cdot x \geq \theta$. Such a pair (w, θ) is said to represent f. See [7, 21] for extensive treatments of linear threshold functions on $\{0,1\}^n$.

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 $w = (0, \dots, 0)$ 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.

The well-known Perceptron convergence theorem bounds the number of mistakes which the Perceptron algorithm can make.

THEOREM 2.1 (see [6, 20, 24]). *Let $\langle x^1, y_1\rangle, \dots, \langle x^m, y_m\rangle$ be a sequence of labeled examples with $\|x^i\| \leq R$ and $y_i \in \{-1,1\}$ for all i. Let u be a vector and θ, ξ be such that $y_i(u \cdot x^i - \theta) \geq \xi$ for all i, where $\xi > 0$. Then the total number of mistakes made by the Perceptron algorithm on this example sequence is at most*

$$\frac{(R^2+1)(\|u\|^2+\theta^2)}{\xi^2}.$$

2.2. Winnow.

The Winnow algorithm takes as input an initial vector $w^I \in \mathbb{R}^n$, a promotion factor $\alpha \in \mathbb{R}$, and a threshold $\theta \in \mathbb{R}$. 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 capable only of expressing positive threshold functions as its hypotheses. This limitation can be easily overcome, however, by using various simple transformations on the input (see [14, 15]). Littlestone has proved the following result, analogous to the Perceptron convergence theorem, bounding the number of mistakes which Winnow makes.

THEOREM 2.2 (see [16]). *Let $\langle x^1, y_1\rangle, \ldots, \langle x^m, y_m\rangle$ be a sequence of labeled examples with $x^i \in [0,1]^n$ and $y_i \in \{-1, 1\}$ for all i. Let u be a positive vector and $\delta > 0$ be such that whenever $y_i = 1$ we have $u \cdot x \geq 1$ and whenever $y_i = -1$ we have $u \cdot x \leq 1 - \delta$. If Winnow is run on this example sequence with initial parameters w^I, α, θ where $w^I = (1, \ldots, 1)$, $\alpha = 1 + \frac{\delta}{2}$ and $\theta > 0$, then the total number of mistakes made by Winnow on this example sequence is at most*

$$\frac{8n}{\delta^2\theta} + \max\left\{0, \frac{14}{\delta^2}\sum_{i=1}^{n} u_i \ln(u_i\theta)\right\}.$$

2.3. PAC learning with online learning algorithms. In Valiant's PAC learning model [12, 26] 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 2.3. *We say that an online learning algorithm A (such as Perceptron or Winnow) is an efficient PAC learning algorithm for concept class C over X 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 X_n, any $c \in C_n$, and any $0 < \epsilon, \delta < 1$:*

- *Given any example $x \in X_n$ 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 said to be an efficient PAC learning algorithm under a fixed distribution $\mathcal{D}$ if it satisfies the above definition for the fixed distribution $\mathcal{D}$. The most natural distribution over a finite set is of course the uniform distribution; we write $\mathcal{U}_n$ to denote the uniform distribution over $\{0, 1\}^n$.

2.4. PAC learning versus online mistake bound learning. Several generic techniques are known [1, 11, 14] for converting any online mistake bound learning algorithm to a PAC algorithm. These conversion procedures yield PAC learning algorithms whose running time is polynomially related to the running time of the original online algorithm. Using these conversion procedures Theorems 2.1 and 2.2 imply that Perceptron and Winnow are efficient PAC learning algorithms for certain restricted linear threshold learning problems. For example, one straightforwardly obtains the following.

COROLLARY 2.4. *Let C_W be the class of linear threshold functions $w \cdot x \geq \theta$ over $\{0,1\}^n$ such that each w_i is an integer and $\sum_{i=1}^n |w_i| < W$. Then both Perceptron and Winnow yield PAC learning algorithms for C_W which run in time poly$(n, W, \frac{1}{\epsilon}, \frac{1}{\delta})$.*

It is well known, though, that there are linear threshold functions over $\{0,1\}^n$ which require integer coefficients of magnitude $2^{\Omega(n)}$. (See, e.g., [10]; we will see an example of such a linear threshold function in section 5.1.) For functions such as these the time bound of Corollary 2.4 is exponentially large in n and hence does not shed light on whether or not Perceptron and Winnow are efficient PAC learning algorithms.

3. Perceptron can learn nested functions under $\mathcal{U}_n$.

In this section we establish a sufficient condition for a family of threshold functions to be efficiently learnable by Perceptron under $\mathcal{U}_n$. We prove that nested functions satisfy this condition and thereby immediately obtain the main result of this section. This complements Schmitt's result [25] that the Perceptron algorithm cannot efficiently PAC learn nested functions under arbitrary distributions.

The class of nested functions, denoted NF_n, was introduced by Anthony, Brightwell, and Shawe-Taylor in [3].

DEFINITION 3.1. *The class of nested functions over $x_1, \ldots, x_n$ 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 3.2. *Any $f \in NF_n$ can be represented by a linear threshold function*

$$w_1 x_1 + \cdots + w_n x_n \geq \theta_n$$

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 $w_1 x_1 + \cdots + w_{n-1} x_{n-1} \geq \theta_{n-1}$, 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 $w_1 x_1 + \cdots + w_{n-1} x_{n-1} + 2^{n-1} x_n \geq \theta_n = \theta_{n-1} + 2^{n-1}$.
2. $f = g \wedge \overline{x}_n$: then f is $w_1 x_1 + \cdots + w_{n-1} x_{n-1} - 2^{n-1} x_n \geq \theta_n = \theta_{n-1}$.
3. $f = g \vee x_n$: then f is $w_1 x_1 + \cdots + w_{n-1} x_{n-1} + 2^{n-1} x_n \geq \theta_n = \theta_{n-1}$.
4. $f = g \vee \overline{x}_n$: then f is $w_1 x_1 + \cdots + w_{n-1} x_{n-1} - 2^{n-1} x_n \geq \theta_n = \theta_{n-1} - 2^{n-1}$.

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. $\square$

DEFINITION 3.3. *Let G_n be a collection of hyperplanes in $\mathbb{R}^n$. 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 Euclidean distance τ of the hyperplane. A class of linear 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 linear threshold function f.*

LEMMA 3.4. *The class of nested functions is gradual.*

Proof. We use the representation established in Lemma 3.2; 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 $m \geq \frac{1}{2}$. The lemma follows by noting that $\|w\| = (\frac{4^n-1}{3})^{1/2} = \Theta(2^n)$ and that the distance from a point x' to the hyperplane $w \cdot x = \theta$ is $\|w\|^{-1} \cdot |w \cdot x' - \theta|$. ☐

This lemma ensures that relatively few points can lie close 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 3.5. *If C is a gradual class of linear threshold functions, then Perceptron is an efficient PAC learning algorithm for C under $\mathcal{U}_n$.*

Proof. The proof is similar to the proof of Theorem 1 in [5]. Let $w \cdot x \geq \theta$ be a gradual 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 3.3, there is some constant $k > 0$ such that for all $\tau > 0$ the probability that a random example drawn from $EX(c, \mathcal{U}_n)$ 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 $EX(c, \mathcal{U}_n)$ is within distance $k\epsilon$ of the hyperplane. Let $B \subseteq \{0,1\}^n$ be the set of examples x which lie within distance $k\epsilon$ of the hyperplane; so $\Pr_{x \in \mathcal{U}_n}[x \in B] \leq \epsilon/2$.

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

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

with $A = \mp(w \cdot x - \theta)$. Since x was misclassified, we know that $\pm(w_t \cdot x - \theta_t) \leq 0$, and hence $\Delta N(\alpha) \leq 2\alpha A + n + 1$. If $x \in B$, then $A \leq 0$; if $x \notin B$, then $A \leq -k\epsilon\alpha$. 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.

Let $m = \max\{144 \ln \frac{\delta}{2}, m_1\}$ and consider running the Perceptron algorithm for $2m/\epsilon$ examples. Let $h_1, h_2, \ldots$ denote the hypotheses which are generated by the Perceptron algorithm during the course of its execution on these $2m/\epsilon$ examples. We have that

$$\begin{aligned}
\Pr[\text{each } h_i \text{ has error } > \epsilon] &= \Pr[(\text{each } h_i \text{ has error } > \epsilon)\ \& \\
&\qquad (\text{fewer than } m \text{ updates take place})] \\
&\quad + \Pr[(\text{each } h_i \text{ has error } > \epsilon)\ \& \\
&\qquad\quad (\text{at least } m \text{ updates take place})] \\
&\leq \Pr[(\text{fewer than } m \text{ updates take place}) \mid \\
&\qquad (\text{each } h_i \text{ has error} > \epsilon)] \\
&\quad + \Pr[(\text{at least } m \text{ updates take place}) \mid \\
&\qquad\quad (\text{each } h_i \text{ has error} > \epsilon)].
\end{aligned}$$

To bound the first of these conditional probabilities, we note that conditioned on the event that each h_i has error at least ϵ, the expected number of updates is at least $2m$. A straightforward Chernoff bound [2] shows that this first conditional probability is at most $\delta/2$.

To bound the second conditional probability, we recall that if at least m updates take place, then at least 7/12 of the updates must be on examples in B. However, as noted earlier, if each h_i has error at least ϵ, then for each update the probability that the update is in B is at most 1/2. Another application of Chernoff bounds shows that the second conditional probability is at most $\delta/2$, and the theorem is proved. $\square$

As an immediate corollary of Theorem 2.1 we have that the Perceptron is a PAC learning algorithm for the class of nested functions under the uniform distribution on $\{0,1\}^n$.

4. Perceptron cannot learn H_n under $\mathcal{U}_n$.

A very simple argument suffices to establish this result. We require the following definition.

DEFINITION 4.1 (see [25]). *The* weight complexity *of a linear threshold function f is the smallest positive integer 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 4.2. *The Perceptron algorithm is not an efficient 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 [9] has constructed a linear 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). However, the amount of computation time which a PAC learning algorithm is allowed is only $\text{poly}(n, \frac{1}{\epsilon}) = \text{poly}(n, 2^n) = 2^{O(n)}$. $\square$

5. Winnow cannot learn H_n.

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

THEOREM 5.1. *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, α, θ) 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 [18] have used a counting argument to show that, given any (w^I, α, θ), there exists a target threshold function and example sequence which together cause Winnow(w^I, α, θ) to make exponentially many mistakes, but no explicit construction was known.

5.1. A threshold function with large coefficients and a small specifying set.

The proof of Theorem 3.5 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 [21, 22].)

LEMMA 5.2. *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 all $i \geq 3$ we have that $u_i \geq F_{i-3} u_3$, where F_i is the ith Fibonacci number: $F_0 = 1, F_1 = 1, F_2 = 1, F_3 = 2, F_4 = 3, \ldots$.

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 $u_4 \geq u_3$. Similarly, since $f_5(0,0,1,1,0) = 0$, we find that $u_5 \geq u_3$.

We now suppose that the lemma is true for $k = 1, 2, \ldots, j-1$ 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 $u_1 x_1 + \cdots + u_{n-2} x_{n-2} \geq \theta - u_n$ 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

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

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

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

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

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

By the induction hypothesis, we find that

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

and the lemma is proved. $\square$

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 [3] before stating the next lemma.

DEFINITION 5.3. *Let $c \in H_n$. We say that c is* consistent *with a set $S = \{\langle x^1, b_1\rangle, \langle x^2, b_2\rangle, \ldots, \langle x^m, b_m\rangle\}$ of labelled examples if $c(x^i) = b_i$ for all i. The set S is said to* specify *c* in *H_n if c is the only function in H_n which is consistent with S; we say that such a set is a* specifying set *for c in H_n. The* specification number *of c, denoted $\sigma_n(c)$, is the smallest size of any specifying set for c in H_n.*

The following results are proved in [3].

LEMMA 5.4.

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

We use the results of Lemma 5.4 to show that the function g_n defined below has a small specifying set.

LEMMA 5.5. *Let n be even and let g_n be the linear 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 5.4, we have $\sigma_{n-1}(g_n\downarrow) = n$.

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 5.4 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 5.4 that $\sigma_n(g_n) \leq 5n - 8$. $\square$

5.2. Proof of Theorem 3.5. 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 5.6. *Let n be even and let g_n be as defined in Lemma* 5.5. *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 jth 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 3.5. 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 5.5 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, first note 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 [23].) 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 5.2 and w is positive, by Lemma 5.2 we have that $w_{n-1} \geq F_{n-4} w_3$. Combining this with Lemma 5.6, 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 $\alpha = 1 + \frac{1}{\Omega(\phi^n)}$. However, 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 $w_1^I \leq w_2^I \leq \cdots \leq w_n^I$, 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 can hence 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 $\alpha < (1 + \frac{1}{F_{n-4}})^2$. 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 $\alpha < (1 + \frac{1}{F_{n-4}})^2$, it follows that $\Omega(n\phi^n)$ update steps will be required. Hence no hypothesis consistent with g_n can be achieved in polynomial time, and the theorem is proved. ◻

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.

6. Conclusion. Many questions remain to be answered about the PAC learning ability of simple online algorithms such as Perceptron and Winnow. Perceptron is now known to be a PAC learning algorithm for linear threshold functions under the uniform distribution on the n-dimensional unit sphere and is known not to be a 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 (Winnow) can PAC learn under the uniform distribution and under arbitrary distributions.

The linear threshold function g_n used in our Winnow proof is very similar to a nested function; in particular, both $g_n\!\uparrow$ and $g_n\!\downarrow$ are nested functions. In light of this fact, and of Schmitt's proof that Perceptron is not a PAC learning algorithm for the class of nested functions, it would be interesting to determine whether Winnow is capable of PAC learning the class of nested functions (or, equivalently, the class of decision lists).

Both the Perceptron and Winnow algorithms are *local* in the sense of [27]: each update to a weight w_i depends only on the value of $w \cdot x$, the value of w_i, the value of x_i, and the correct classification of the example. Such algorithms are of particular interest because they require very limited communication between the processors that perform the updates for each weight and are hence well suited for implementation on a distributed system such as a neural network. Known algorithms for learning threshold functions efficiently, such as the algorithm of [18] which is based on linear programming, are nonlocal. We conjecture that local learning algorithms cannot efficiently learn the unrestricted class of threshold functions.

A final issue is attribute efficiency. A learning algorithm is said to be *attribute efficient* if, when the target concept has k relevant variables out of n, the number of calls which the algorithm makes to the example oracle is polynomial in k and poly-logarithmic in n. Littlestone's results for Winnow show that it is an attribute efficient algorithm for certain simple subclasses of threshold functions such as disjunctions and r-out-of-k threshold functions. Our results imply that Winnow is not an attribute efficient PAC learning algorithm for the unrestricted class of linear threshold functions. It would be interesting to gain a better understanding of the abilities and limitations of Winnow as an attribute efficient learning algorithm.

Acknowledgment. We thank Les Valiant for helpful comments and suggestions.

REFERENCES

[1] D. Angluin, *Queries and concept learning,* Machine Learning, 2 (1988), pp. 319–342.
[2] D. Angluin and L. G. Valiant, *Fast probabilistic algorithms for Hamiltonian circuits and matchings,* J. Comput. System Sci., 18 (1979), pp. 155–193.
[3] M. Anthony, G. Brightwell, and J. Shawe-Taylor, *On specifying boolean functions using labelled examples,* Discrete Appl. Math., 61 (1993), pp. 1–25.
[4] P. Auer and M. Warmuth, *Tracking the best disjunction,* in Proceedings of the 36th Annual Symposium on Foundations of Computer Science, IEEE Computer Society Press, Los Alamitos, CA, 1995, pp. 312–321.
[5] E. B. Baum, *The perceptron algorithm is fast for nonmalicious distributions,* Neural Comput., 2 (1990), pp. 248–260.
[6] H. Block, *The perceptron: A model for brain functioning,* Rev. Modern Phys., 34 (1962), pp. 123–135.
[7] M. L. Dertouzos, *Threshold Logic: A Synthesis Approach,* MIT Press, Cambridge, MA, 1965.
[8] Y. Freund and R. Schapire, *Large margin classification using the Perceptron algorithm,* in Proceedings of the 11th Annual Conference on Computational Learning Theory, ACM, New York, 1998, pp. 209–217.
[9] J. Håstad, *On the size of weights for threshold gates,* SIAM J. Discrete Math., 7 (1994), pp. 484–492.
[10] S. Hampson and D. Volper, *Linear function neurons: Structure and training,* Biol. Cybernet., 53 (1986), pp. 203–217.
[11] D. Haussler, *Space Efficient Learning Algorithms,* Technical Report UCSC-CRL-88-2, University of California, Santa Cruz, CA, 1988.
[12] M. Kearns and U. Vazirani, *An Introduction to Computational Learning Theory,* MIT Press, Cambridge, MA, 1994.
[13] J. Kivinen, M. Warmuth, and P. Auer, *The perceptron algorithm vs. winnow: Linear vs. logarithmic mistake bounds when few input variables are relevant,* in Proceedings of the 8th Annual Conference on Computational Learning Theory, ACM, New York, 1995, pp. 289–296.
[14] N. Littlestone, *Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm,* Machine Learning, 2 (1988), pp. 285–318.
[15] N. Littlestone, *Mistake Bounds and Logarithmic Linear-Threshold Learning Algorithms,* Ph.D. thesis, Technical Report UCSC-CRL-89-11, University of California, Santa Cruz, CA, 1989.
[16] N. Littlestone, *Redundant noisy attributes, attribute errors, and linear-threshold learning using winnow,* in Proceedings of the 4th Annual Conference on Computational Learning Theory, ACM, New York, 1991, pp. 147–156.
[17] W. Maass and M. Warmuth, *Efficient Learning with Virtual Threshold Gates,* Technical Report UCSC-CRL-95-37, University of California, Santa Cruz, CA, 1995.
[18] 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, and R. Rivest, eds., MIT Press, Cambridge, MA, 1994, pp. 381–414.
[19] M. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry,* expanded ed., MIT Press, Cambridge, MA, 1988.
[20] A. Novikoff, *On convergence proofs on perceptrons,* in Proceedings of the Symposium on Mathematical Theory of Automata, Vol. 12, 1962, pp. 615–622.
[21] S. Muroga, *Threshold Logic and its Applications,* Wiley-Interscience, New York, 1971.
[22] I. Parberry, *Circuit Complexity and Neural Networks,* MIT Press, Cambridge, MA, 1994.
[23] L. Pitt and L. G. Valiant, *Computational limitations on learning from examples,* J. ACM, 35 (1988), pp. 965–984.

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

[25] M. Schmitt, *Identification criteria and lower bounds for Perceptron-like learning rules,* Neural Comput., 10 (1998), pp. 235–250.

[26] L. G. Valiant, *A theory of the learnable,* Comm. ACM, 27 (1984), pp. 1134–1142.

[27] L. G. Valiant, *Circuits of the Mind,* Oxford University Press, New York, 1994.

[28] L. G. Valiant, *Projection learning,* in Proceedings of the 11th Annual Conference on Computational Learning Theory, ACM, New York, 1998, pp. 287–293.