A Polynomial-time Algorithm for Learning Noisy Linear Threshold Functions[*]

Avrim Blum[†] Alan Frieze[‡] Ravi Kannan[§] Santosh Vempala[¶]

Abstract

In this paper we consider the problem of learning a linear threshold function (a half-space in n dimensions, also called a "perceptron"). Methods for solving this problem generally fall into two categories. In the absence of noise, this problem can be formulated as a Linear Program and solved in polynomial time with the Ellipsoid Algorithm or Interior Point methods. Alternatively, simple greedy algorithms such as the Perceptron Algorithm are often used in practice and have certain provable noise-tolerance properties; but, their running time depends on a separation parameter, which quantifies the amount of "wiggle room" available for a solution, and can be exponential in the description length of the input.

In this paper, we show how simple greedy methods can be used to find weak hypotheses (hypotheses that correctly classify noticeably more than half of the examples) in polynomial time, *without* dependence on any separation parameter. Suitably combining these hypotheses results in a polynomial-time algorithm for learning linear threshold functions in the PAC model in the presence of random classification noise. (Also, a polynomial-time algorithm for learning linear threshold functions in the Statistical Query model of Kearns.)

Our algorithm is based on a new method for removing outliers in data. Specifically, for any set S of points in $\boldsymbol{R}^n$, each given to b bits of precision, we show that one can remove only a small fraction of S so that in the remaining set T, for every vector v, $\max_{x \in T}(v \cdot x)^2 \leq poly(n, b)\mathbf{E}_{x \in T}(v \cdot x)^2$; i.e., for any hyperplane through the origin, the *maximum* distance (squared) from a point in T to the plane is at most polynomially larger than the average. After removing these outliers, we are able to show that a modified version of the Perceptron Algorithm finds a weak hypothesis in polynomial time, even in the presence of random classification noise.

1 Introduction

The problem of learning a linear threshold function is one of the oldest problems studied in machine learning. Typically, this problem is solved by using simple greedy methods. For

[*]An earlier version of this paper appeared in the 37th Symp. on Foundations of Computer Science, 1996.

[†]School of Computer Science, Carnegie Mellon University, Pittsburgh PA 15213. Supported in part by NSF National Young Investigator grant CCR-9357793, a Sloan Foundation Research Fellowship, and by ARPA under grant F33615-93-1-1330. Email: `avrim@cs.cmu.edu`.

[‡]Department of Mathematical Sciences, Carnegie Mellon University, Pittsburgh PA 15213. Supported in part by NSF grant CCR9225008. Email: `af1p@andrew.cmu.edu`.

[§]School of Computer Science, Carnegie Mellon University, Pittsburgh PA 15213. Supported in part by NSF grant CCR9528973. Email: `kannan+@cs.cmu.edu`.

[¶]School of Computer Science, Carnegie Mellon University, Pittsburgh PA 15213. Supported in part by NSF National Young Investigator grant CCR-9357793. Email: `svempala@cs.cmu.edu`.

instance, one commonly-used greedy algorithm for this task is the Perceptron Algorithm [Ros62, Agm54], described below in Section 3. These algorithms have running times that depend on the amount of "wiggle room" available to a solution. In particular, the Perceptron Algorithm has the following guarantee [MP69]. Given a collection of data points in $\boldsymbol{R}^n$, each labeled as *positive* or *negative*, the algorithm will find a vector w such that $w \cdot x > 0$ for all positive points x and $w \cdot x < 0$ for all negative points x, if such a vector exists.[1] Moreover, the number of iterations made by the algorithm is at most $1/\sigma^2$ where σ is a "separation parameter" defined as the largest value such that for some vector w^*, all positive x satisfy $\cos(w^*, x) > \sigma$, and all negative x satisfy $\cos(w^*, x) < -\sigma$, where $\cos(a, b) = \frac{a \cdot b}{|a||b|}$ is the cosine of the angle between vectors a and b.

Unfortunately, it is possible for the separation parameter σ to be exponentially small, and for the algorithm to take exponential time, even if all the examples belong to $\{0,1\}^n$. A classic setting in which this can occur is a data set labeled according to the function "if $x_1 = 1$ then positive else if $x_2 = 1$ then negative else if $x_3 = 1$ then positive, ...". This function *has* a linear threshold representation, but it requires exponentially large weights and can cause the Perceptron Algorithm to take exponential time. (In practice, though, the Perceptron Algorithm and its variants tend to do fairly well; e.g., see [AR88].)

Given this difficulty, one might propose instead to use a polynomial-time linear programming algorithm to find the desired vector w. Each example provides one linear constraint and one could simply apply an LP solver to solve them [Kha79, Kar84, MT89]. In practice, however, this approach is less often used in machine learning applications. One of the main reasons is that the data often is not consistent with *any* vector w and one's goal is simply to do as well as one can. And, even though finding a vector w that minimizes the number of misclassified points is NP-hard, variants on the Perceptron Algorithm typically do well in practice[Gal90, Ama94]. In fact, it is possible to provide guarantees for variations on the Perceptron Algorithm in the presence of inconsistent data (e.g., see [Byl93, Byl94, Kea93][2]), under models in which the inconsistency is produced by a sufficiently "benign" process, such as the *random classification noise model* discussed below.

In this paper, we present a version of the Perceptron Algorithm that maintains its properties of noise-tolerance, while providing polynomial-time guarantees. Specifically, the algorithm we present is guaranteed to provide a weak hypothesis (one that correctly classifies noticeably more than half of the examples) in time polynomial in the description length of the input and *not* dependent on any separation parameter. The output produced by the algorithm can be thought of as a "thick hyperplane," satisfying the following two properties:

1. Points outside of this thick hyperplane are classified with high accuracy (points inside can be viewed as being classified as "I don't know").

2. At least a $1/poly$ fraction of the input distribution lies outside of this hyperplane.

This sort of hypothesis can be easily boosted in a natural way (by recursively running the algorithm on the input distribution restricted to the "don't know" region) to achieve a hypothesis of arbitrarily low error.[3] This yields the following theorem.

[1]If a non-zero threshold is desired, this can be achieved by adding one extra dimension to the space.

[2]The word "polynomial" in the title of [Byl93] means polynomial in the inverse of the separation parameter, which as noted above can be exponential in n even when points are chosen from $\{0,1\}^n$.

[3]Thanks to Rob Schapire for pointing out that standard Boosting results [Sch90, Fre92] do not apply in the

Theorem 1 *The class of linear threshold functions in $\boldsymbol{R}^n$ can be learned in polynomial time in the PAC prediction model in the presence of random classification noise.*

Remark: The learning algorithm can be made to fit the Statistical Query learning model [Kea93].

The main idea of our result is as follows. First, we modify the standard Perceptron Algorithm to produce an algorithm that succeeds in weak learning unless an overwhelming fraction of the data points lie on or very near to some hyperplane through the origin. Specifically, the algorithm succeeds unless there exists some "bad" vector w such that most of the data points x satisfy $|\cos(w,x)| < \delta$ for some small $\delta > 0$. Thus, we are done if we can somehow preprocess the data to ensure that no such bad vector w exists.

The second part of our result is a method for appropriately preprocessing the data. One natural approach that *almost* works is to use the principal components of the data set S to perform a linear transformation so that for every hyperplane through the origin, the average squared distance of the examples to the hyperplane is 1. In other words, for every unit vector w, we now have $\mathbf{E}_{x\in S}(w\cdot x)^2 = 1$. (This assumes that there are no planes through the origin on which *all* the examples lie, but that case is easy to handle by restricting to that plane and reducing n by 1.) Unfortunately, it is possible that this linear transformation will not solve our problem because of the presence of a small number of outliers. For instance, there may exist a unit vector w such that even though the *average* value of $(w\cdot x)^2$ is 1, almost all of the points x satisfy $w\cdot x = 0$, and just a few outliers have a very large dot product with w.

We solve this last problem by proving the following result. Given any set S of points in n dimensional space, each requiring b bits of precision, one can remove only a small fraction of those points and then guarantee that in the set T remaining, for every vector v,

$$\max_{x\in T}(x\cdot v)^2 \leq \mathit{poly}(n,b)\mathbf{E}_{x\in T}[(x\cdot v)^2].$$

In this sense, the set remaining has no outliers with respect to any hyperplane through the origin. In addition, we show that removing these outliers can be done in polynomial time. After removing these outliers, we can then apply the linear transformation mentioned above so that in the transformed space, for every unit vector v,

$$\mathbf{E}_{x\in T}[(x\cdot v)^2] = 1 \quad \textit{and} \quad \max_{x\in T}(x\cdot v)^2 \leq \mathit{poly}(n,b).$$

Because the maximum is bounded, having the expectation equal to 1 means that for every hyperplane through the origin, at least a $1/\mathit{poly}(n,b)$ fraction of the examples are at least a $1/\mathit{poly}(n,b)$ distance away, which then allows us to guarantee that the modified Perceptron Algorithm will be a weak learner.

context of random classification noise. (It is an open question whether arbitrary weak-learning algorithms can be boosted in the random classification noise model.) Thus, we use the fact that the hypothesis produced by the algorithm can be viewed as a high-accuracy hypothesis over a known, non-negligible portion of the input distribution. Alternatively, Aslam and Decatur [AD93] have shown that Statistical Query (SQ) algorithms *can*, in fact, be boosted in the presence of noise. Since our algorithm can be made to fit the SQ framework (see Section 4.1), we could also apply their results to achieve strong learning.

1.1 The structure of this paper

We will begin by formally stating the Outlier Removal Lemma, whose proof is deferred to a later section. We then consider the problem of learning a linear threshold function in the case of zero noise and describe how the Perceptron Algorithm can be modified and combined with the procedure from the Outlier Removal Lemma to produce a polynomial time PAC-learning algorithm. Finally, we describe how the algorithm can be adjusted to the noisy case using known techniques [Byl94, Kea93, AD94].

1.2 Notation, definitions, and preliminaries

In this paper, we consider the problem of learning linear threshold functions in the PAC model in the presence of random classification noise [KV94]. The problem can be stated as follows.

We are given access to examples (points) drawn from some distribution D over R^n. Each example is labeled as positive or negative. The labels on examples are determined by some unknown target function $w^* \cdot x > 0$ (i.e., x is positive if $w^* \cdot x > 0$ and is negative otherwise) but each label is then flipped independently with some fixed probability $\eta < 1/2$ before it is presented to the algorithm. η is called the *noise rate*. We assume that all points are given to some b bits of precision. More precisely, we define $I_b = \{p/q : |p|, |q| \in \{0, 1, 2, \ldots, 2^b - 1\}, q \neq 0\}$, and assume that D is restricted to I_b^n (i.e., $I_b \times \cdots \times I_b$).

A *hypothesis* is a polynomial-time computable function. The error of a hypothesis h with respect to the target function is the probability that h disagrees with the target function on a random example drawn from D. Thus, if h has error ϵ, then the probability for a random x that $h(x)$ disagrees with the noisy label observed is $(1-\eta)\epsilon + \eta(1-\epsilon) = \eta + \epsilon(1-2\eta)$.

Our goal is an algorithm that for any (unknown) distribution D, any (unknown) target concept $w^* \cdot x > 0$, any (unknown) $\eta < 1/2$, and any inputs $\epsilon, \delta > 0$, with probability at least $1-\delta$ produces a hypothesis whose error with respect to the target function is at most ϵ. The algorithm may request a number of examples polynomial in $n, b, 1/\epsilon, \log(\frac{1}{\delta})$, and $\frac{1}{1-2\eta}$, and should run in time polynomial in these parameters as well.

The algorithms we describe are most easily viewed as working with a fixed sample of data. We can apply the algorithms to the PAC setting by running them on a sufficiently large sample of data drawn according to the above model, and then applying standard VC-dimension arguments to the result [VC71].

For most of this paper, we will consider the above problem for the case of zero noise ($\eta = 0$), which we extend to the general case in Section 4. The reason for considering the $\eta = 0$ case first is that we will be modifying algorithms that have already been proven tolerant to random classification noise (e.g., [Byl94]), and the key issue is getting the polynomial time guarantee. The extension to $\eta \neq 0$ is a bit messy, but follows well-trodden ground.

2 The Outlier Removal Lemma

Our main lemma, needed for our algorithm and analysis, states that given any set of data points in I_b^n, one can remove a small portion and guarantee that the remainder contains no outliers in a certain well-defined sense.

Lemma 1 (Outlier Removal Lemma) *For any set $S \subseteq I_b^n$ and any $\varepsilon > 0$, there exists a subset $S' \subseteq S$ such that:*

(i) $|S'| \geq (1 - \varepsilon - 2^{-nb})|S|$*, and*

(ii) for every vector $w \in \boldsymbol{R}^n$, $\max_{x \in S'}(w \cdot x)^2 \leq \beta \mathbf{E}_{x \in S'}[(w \cdot x)^2]$,

where $\beta = O(n^7 b/\varepsilon)$*. Moreover, such a set S' can be computed in polynomial time.*

It turns out that the algorithm for computing the set S' of Lemma 1 is quite simple and in fact shares many characteristics with the high-level description given earlier in Section 1 of how the Lemma is used. The algorithm is as follows:

First, we may assume that the matrix X of points in S has rank n; otherwise we simply drop to the subspace spanned. Next we perform a linear transformation so that in the transformed space, for every unit vector w, $\mathbf{E}_{x \in S}[(w \cdot x)^2] = 1$. This transformation is just left-multiplication by A^{-1} (so the new set of points is $A^{-1}X$) where A^2 is the symmetric factorization of XX^T that can be determined by an eigenvalue/eigenvector computation. Next we remove all points $x \in S$ such that $|x|^2 \geq \beta/144n$. If S now satisfies the condition of the theorem we stop. Otherwise, we repeat.

The difficult issue is proving that this algorithm will in fact halt before removing too many points from S. The proof of this fact is deferred to Section 5.

3 The Perceptron Algorithm

The Perceptron Algorithm[Ros62, Agm54] operates on a set S of labeled data points in n dimensional space. Its goal is to find a vector w such that $w \cdot x > 0$ for all positive points x and $w \cdot x < 0$ for all negative points x. We will say that such a vector w *correctly classifies* all points in S. If a non-zero threshold value is desired, this can be handled by simply creating an extra $(n+1)$st coordinate and giving all examples a value of 1 in that coordinate.

For convenience, define $\ell(x)$ (the label of x) to be 1 if x is positive and -1 if x is negative. So, our goal is to find a vector w such that $\ell(x)(w \cdot x) > 0$ for all $x \in S$. Also, for a point x let $\hat{x} = x/|x|$. I.e., $\hat{x}$ is the vector x normalized to have length 1.

3.1 The standard algorithm

The standard algorithm proceeds as follows. We begin with $w = \vec{0}$. We then perform the following operation until all examples are correctly classified:

Pick some arbitrary misclassified example $x \in S$ and let $w \leftarrow w + \ell(x)\hat{x}$.

A classic theorem (see [MP69]) describes the convergence properties of this algorithm.

Theorem 2 *[MP69] Suppose the data set S can be correctly classified by some unit vector w^*. Then, the Perceptron Algorithm converges in at most $1/\sigma^2$ iterations, where $\sigma = min_{x \in S}|w^* \cdot \hat{x}|$.*

Proof. Consider the cosine of the angle between the current vector w and the unit vector w^* given in the theorem. That is, $\frac{w \cdot w^*}{|w|}$. In each step of the algorithm, the numerator of this fraction increases by at least σ because $(w+\ell(x)\hat{x})\cdot w^* = w\cdot w^* + \ell(x)\hat{x}\cdot w^* \geq w\cdot w^* + \sigma$. On the other hand, the square of the denominator increases by at most 1 because $|w+\ell(x)\hat{x}|^2 = |w|^2 + 2\ell(x)(w\cdot\hat{x}) + 1 < |w|^2 + 1$ (since x was misclassified, this means the crossterm is negative). Therefore, after t iterations, $w \cdot w^* \geq t\sigma$ and $|w| < \sqrt{t}$. Notice that the former cannot be larger than the latter. Thus, $t \leq 1/\sigma^2$. □

3.2 A modified version

We now describe a modified version of the Perceptron Algorithm that will be needed for our construction. Recall our notation that $\cos(a, b)$ is the cosine of the angle between vectors a and b, or equivalently $\frac{a\cdot b}{|a||b|}$.

The reason we need to modify the algorithm is this: In the standard algorithm, if some of the points are far from the target plane (in the sense that $\cos(w^*, x)$ is large) and some are near, then eventually the hypothesis will correctly classify the far away points but may make mistakes on the nearby ones. This is simply because the points far from $w^* \cdot x = 0$ cause the algorithm to make substantial progress but the others do not. Unfortunately, we cannot test for points being far or near to the target plane. So, we cannot produce the rule: "if $|\cos(w^*, x)|$ is large then predict based on $x \cdot w$, else say 'I don't know'." What we want instead is an algorithm that does well on points that are far from the *hypothesis* plane, because $|\cos(w, x)|$ is something that the algorithm *can* calculate. If we then can guarantee that a reasonable fraction of points will have this property, we will have our desired weak hypothesis (just replacing w^* by w in the above rule).

Specifically, our modified algorithm takes as input a quantity σ and its goal is to produce a vector w such that every misclassified $x \in S$ should satisfy $|\cos(w, x)| \leq \sigma$. The algorithm proceeds as follows.

The Modified Perceptron Algorithm

1. Begin with w as a random unit vector.

2. If every misclassified $x \in S$ satisfies $|\cos(w, x)| \leq \sigma$ (i.e., if $|w \cdot \hat{x}| \leq \sigma|w|$) then halt.

3. Otherwise, pick the misclassified $x \in S$ maximizing $|\cos(w, x)|$ and update w using:

$$w \leftarrow w - (w \cdot \hat{x})\hat{x}.$$

   In other words, we add to w the appropriate multiple of x so that w is now orthogonal to x, i.e., we add the multiple of x that shrinks w as much as possible.

4. If we have made fewer than $(1/\sigma^2)\ln n$ updates then go back to Step 2. Otherwise, go back to Step 1 (begin anew with a new random unit starting vector).

Theorem 3 *If the data set S is linearly separable, then with probability $1 - \delta$ the modified perceptron algorithm halts after $O((1/\sigma^2)\ln(n)\ln(\frac{1}{\delta}))$ iterations, and produces a vector w such that every misclassified $x \in S$ satisfies $|\cos(w, x)| \leq \sigma$.*

Proof. Let w^* be a unit vector that correctly classifies all $x \in S$. Suppose it is the case that the initial (random unit) vector w satisfies $w \cdot w^* \geq 1/\sqrt{n}$. Notice that in each update made in Step (3), $w \cdot w^*$ does not decrease because

$$(w - (w \cdot \hat{x})\hat{x}) \cdot w^* = w \cdot w^* - (w \cdot \hat{x})(w^* \cdot \hat{x}) \geq w \cdot w^*$$

where the last inequality holds because w misclassifies x. On the other hand, $|w|^2$ *does* decrease significantly because (this is just the Pythagorean Theorem)

$$\begin{aligned} |(w - (w \cdot \hat{x})\hat{x})|^2 &= |w|^2 - 2(w \cdot \hat{x})^2 + (w \cdot \hat{x})^2 \\ &\leq |w|^2(1 - \sigma^2). \end{aligned}$$

Thus, after t iterations, $|w| \leq (1 - \sigma^2)^{t/2}$. Since $|w|$ cannot be less than $w \cdot w^*$, this means that the number of iterations t satisfies $(1 - \sigma^2)^{t/2} \geq 1/\sqrt{n}$, which implies $t \leq (\ln n)/\sigma^2$.

Each time we choose a random initial unit vector for w, there is at least a constant > 0 probability that w satisfies our desired condition that $w \cdot w^* > 1/\sqrt{n}$. Thus, the theorem follows. □

We have described the algorithm as one that runs in *expected* polynomial time. Alternatively we could stop the algorithm after a suitable number of iterations and have a high probability of success. In Section 4 we will alter this algorithm slightly to make it tolerant to random classification noise.

3.3 Combining the Perceptron Algorithm with the removal of outliers

The Modified Perceptron Algorithm can be combined with the Outlier Removal Lemma in a natural way. Given a data set S, we use the Lemma to produce a set S' with $|S'| \geq \frac{1}{2}|S|$ and such that for all vectors w, $\max_{S'}(w \cdot x)^2 \leq \beta \mathbf{E}_{S'}[(w \cdot x)^2]$ where β is polynomial in n and b.

We then reduce dimensionality if necessary to get rid of any vectors w for which the above quantity is zero. That is, we project onto the subspace L spanned by the eigenvectors of the XX^T matrix having non-zero eigenvalue (X is the matrix of points in S'). Now, we perform the linear transformation A^{-1} described in Section 2 so that in the transformed space, for all unit vectors w, $\mathbf{E}_{S'}[(w \cdot x)^2] = 1$. Our guarantee for set S' implies that in the transformed space, $\max_{S'} |x|^2 \leq \beta n$. Thus, for any unit vector w,

$$\begin{aligned} \mathbf{E}_{S'}[\cos(w, x)^2] &= \mathbf{E}_{S'} \frac{(w \cdot x)^2}{|x|^2} \\ &\geq \frac{\mathbf{E}_{S'}[(w \cdot x)^2]}{\max_{S'} |x|^2} \\ &\geq 1/(\beta n). \end{aligned}$$

This implies that in the transformed space, at least a $1/(2\beta n)$ fraction of the points in S' satisfy $\cos(w, x)^2 \geq 1/(2\beta n)$. We can now run the Modified Perceptron Algorithm with $\sigma = 1/\sqrt{2\beta n}$, and guarantee that at the end, at least a $1/(2\beta n)$ fraction of the points in S' satisfy $|\cos(w, x)| \geq \sigma$.

The final hypothesis of the algorithm, in the original untransformed space, is: if $x \notin L$ or $|cos(w, A^{-1}x)| < \sigma$ then guess the label randomly (or say "I don't know"), and otherwise predict according to the hypothesis $w^T A^{-1} x > 0$.

3.4 Achieving Strong (PAC) Learning

The algorithm presented above splits the input space into a classification region

$$\{x : x \in L \text{ and } |\cos(w, A^{-1}x)| \geq \sigma\}$$

and a don't-know region

$$\{x : x \notin L \text{ or } |\cos(w, A^{-1}x)| < \sigma\}.$$

By standard VC-dimension arguments [VC71], if the sample S is drawn from distribution D, then for any $\epsilon, \delta > 0$, if S is sufficiently (polynomially) large, then with high probability ($\geq 1 - \delta$), the true error of the hypothesis inside the classification region is less than ϵ. Furthermore, the weight under D of the classification region is at least $1/poly(n, b)$; that is, the fraction of S that lies in the classification region is representative of the weight of this region under D. Therefore, we can boost the accuracy of the learning algorithm by simply running it recursively on the distribution D restricted to the don't-know region. The final hypothesis produced by this procedure is a decision list of the form: "if the example lies in the classification region of hypothesis 1, then predict using hypothesis 1, else if the example lies in the classification region of hypothesis 2, then predict using hypothesis 2, and so on".

4 Learning with Noise

We now describe how the Modified Perceptron Algorithm can be converted to one that is robust to *random classification noise.* We present two ways of doing this. The first is to recast the algorithm in the Statistical Query (SQ) model of Kearns [Kea93] as extended by Aslam and Decatur [AD94], and to use the fact that any SQ algorithm can be made tolerant of random classification noise. The second is a direct argument along the lines of Bylander [Byl94], who describes how the standard Perceptron Algorithm can be modified to work in this noise model.

We begin with some observations needed for both approaches. For convenience, in the discussion below we will normalize the examples to all have length 1, so that we need not distinguish between x and $\hat{x}$. Recall that $\ell(x) = 1$ if x is a positive example and $\ell(x) = -1$ if x is a negative example.

The first observation is that the only properties of the point x selected in Step 3 of the Modified Perceptron Algorithm that are actually used in the analysis of Theorem 3 are:

$$\begin{aligned} \cos(w, x)\ell(x) &\leq -\sigma, \text{ and} && (1)\\ \cos(w^*, x)\ell(x) &\geq 0. && (2) \end{aligned}$$

The second observation is that, in fact, we only need points that *approximately* achieve these two properties. In particular, suppose that every point x we use in Step 3 satisfies the relaxed conditions:

$$\begin{aligned} \cos(w, x)\ell(x) &\leq -\sigma/2, \text{ and} && (3)\\ \cos(w^*, x)\ell(x) &\geq \frac{-\sigma^2}{16\sqrt{n}\ln n}. && (4) \end{aligned}$$

The first condition guarantees that after $t = (8 \ln n)/\sigma^2$ iterations we have $|w| \leq (1 - (\sigma/2)^2)^{t/2} < 1/n$. The second guarantees that if initially $w \cdot w^* \geq \frac{1}{\sqrt{n}}$, then after t iterations $w \cdot w^* \geq \frac{1}{\sqrt{n}} - \frac{t\sigma^2}{16\sqrt{n}\ln n} \geq \frac{1}{2\sqrt{n}}$. Therefore, we are guaranteed to halt before t iterations have been made.

The final observation is that any positive multiple of

$$\mu_{w,S} = \mathbf{E}_S[\ell(x)x : \cos(w, x)\ell(x) \leq -\sigma]$$

will satisfy conditions (1) and (2), assuming zero noise so that every $x \in S$ satisfies (2), and if we define $\ell(\mu_{w,S}) = 1$. Furthermore, any point sufficiently near to $\mu_{w,S}$ will satisfy the relaxed conditions (3) and (4). Specifically, the definition of $\mu_{w,S}$, the fact that all examples have length 1, and condition (1) together imply that $|\mu_{w,S}| \geq \sigma$. So, any point $\tilde{\mu}_{w,S}$ such that $|\tilde{\mu}_{w,S} - \mu_{w,S}| \leq \sigma^3/(16\sqrt{n}\ln n)$ satisfies conditions (3) and (4).

4.1 Learning with Noise via Statistical queries

Let f be a function from labeled examples to $[0, 1]$. That is, in our setting,

$$f : R^n \times \{-1, 1\} \longrightarrow [0, 1].$$

A *statistical query* is a request for the expected value of f over examples drawn from distribution D and labeled according to the target concept c; i.e., a request for $\mathbf{E}_{x \in D}[f(x, c(x))]$. Assuming that f is polynomial-time computable, it is clear that given access to non-noisy data, this expectation can be estimated to any desired accuracy τ with any desired confidence $1 - \delta$ in time $poly(\frac{1}{\tau}, \log(\frac{1}{\delta}))$, by simply calculating the expectation over a sufficiently large sample. Kearns [Kea93] and Aslam and Decatur [AD94] prove that one can similarly perform such an estimation even in the presence of random classification noise.[4] Specifically, for any noise rate $\eta < 1/2$ and any accuracy (or *tolerance*) parameter τ, the desired expectation can be estimated with confidence $1 - \delta$ in time (and sample size) $poly(\frac{1}{\tau}, \log(\frac{1}{\delta}), \frac{1}{1-2\eta})$. Thus, to prove an algorithm tolerant to random classification noise, it suffices to show that its use of labeled examples can be recast as requests for approximate expectations of this form.

The Modified Perceptron Algorithm uses labeled examples in two places. The first is in Step 2 where we ask if there are any points $x \in S$ such that $\cos(w, x)\ell(x) \leq -\sigma$, and we halt if there are none. We can replace this with a statistical query requesting the probability that a random labeled example from D satisfies this property (formally, a request for $\mathbf{E}_{x \in D}[f(x, c(x))]$ where $f(x, \ell) = 1$ if $\cos(w, x)\ell \leq -\sigma$ and $f(x, \ell) = 0$ otherwise) and halting if this probability is sufficiently small. Specifically, we can set $\tau = \frac{1}{3}\epsilon/(2\beta n)$ and halt if the result of the query is at most $\frac{2}{3}\epsilon/(2\beta n)$, where $1/(2\beta n)$ is a lower bound on $\Pr_{x \in D}(|\cos(w, x)| \geq \sigma)$ from the Outlier Removal Lemma.

The second place that labeled examples are used is in Step 3. As noted in the discussion following equations (3) and (4), it suffices for this step to use a good approximation to $\mu_{w,S}$ instead of using any specific labeled example. We can find such an approximation via statistical queries. Specifically, to approximate the ith coordinate of $\mu_{w,S}$, we ask for $\mathbf{E}_{x \in D}[\ell(x)x_i | \cos(w, x)\ell(x) \leq -\sigma]$. This *conditional* expectation can be approximated

[4]Kearns [Kea93] considers queries with range $\{0, 1\}$. Aslam and Decatur [AD94] extends these arguments (among other things) to queries with range $[0, 1]$, which is more convenient for our purposes.

from statistical queries since we are guaranteed from Step 2 that $\Pr(\cos(w,x)\ell(x) \leq -\sigma)$ is reasonably large. Finally, we combine the approximations for each coordinate into an approximation $\tilde{\mu}_{w,S}$ of $\mu_{w,S}$.

Note that examples are also used in the algorithm for the Outlier Removal Lemma. However, since this algorithm ignores the labels, it is unaffected by random classification noise.

4.2 A direct analysis

We now describe a direct method for making the algorithm noise tolerant, along the lines of Bylander [Byl94]. First, for simplicity, we will reflect negative examples through the origin, and view every example as having a positive label. Thus we can ignore the "$\ell(x)$" term in equations (1)–(4) and in the definition of $\mu_{w,S}$ at the beginning of this section.

We now consider the addition of random noise. Let S denote the original set of non-noisy examples (which our algorithm does not get to see) and let S^η denote the set in which each $x \in S$ independently at random has been reflected through the origin with probability η. I.e., S^η is the noisy data seen by the algorithm. For simplicity, let us assume that η is known to the algorithm; we will see how to remove this assumption at the end of the section.

For a given vector w, define

$$S^\eta_{error} = \{x \in S^\eta : \cos(w,x) < -\sigma\}$$

and

$$S^\eta_{correct} = \{x \in S^\eta : \cos(w,x) > \sigma\}.$$

For convenience, for any set S' define $\text{Sum}[S'] = \sum_{x \in S'} x$. We claim that a quantity that suffices for performing the update of Step 3 is now simply

$$x_{update} = \text{Sum}[S^\eta_{error}] + \frac{\eta}{1-\eta}\text{Sum}[S^\eta_{correct}]. \tag{5}$$

To see why this is a good vector, define $S_{correct} = \{x \in S : \cos(w,x) > \sigma\}$ and $S_{error} = \{x \in S : \cos(w,x) < -\sigma\}$. Let us now, for the purpose of exposition, make the assumption:

> **A:** *The vectors w produced by the algorithm are independent of the noise.* (This is clearly erroneous and it will be removed shortly.)

Then, with respect to the random choice of noisy examples, we have: (noting that the noise does not change $|\cos(w,x)|$)

$$\mathbf{E}[\text{Sum}[S^\eta_{error}]] = (1-\eta)\text{Sum}[S_{error}] - \eta\text{Sum}[S_{correct}]$$

and

$$\mathbf{E}[\text{Sum}[S^\eta_{correct}]] = (1-\eta)\text{Sum}[S_{correct}] - \eta\text{Sum}[S_{error}].$$

Therefore, the expected value of the vector x_{update} used for updating is simply

$$\begin{aligned} \mathbf{E}[x_{update}] &= (1-\eta-\eta^2/(1-\eta))\text{Sum}[S_{error}] \\ &= \left(\frac{1-2\eta}{1-\eta}\right)\text{Sum}[S_{error}], \end{aligned} \tag{6}$$

which is a multiple of the desired vector $\mu_{w,S}$.

We now show that given a sufficiently large sample, with high probability either (Case 1) the calculated value of x_{update} is small implying that the current hypothesis is a good classifier, or else (Case 2) the value of x_{update} calculated is sufficiently close to its expectation to satisfy conditions (3) and (4). In what follows, we use "with high probability" to mean with probability at least $1 - n^{-c}$ where "c" can be increased by adjusting the base of the logarithms to appropriate constants.

We are going to assume that

$$m = |S_{correct} \cup S_{error}| \geq m_0 = \frac{10^4 n^2 (\ln n)^4}{\epsilon^2 \sigma^6} \frac{(1-\eta)^2}{(1-2\eta)^2}.$$

where $\epsilon < 1/2$. ¿From Lemma 1 this amounts to assuming that $|S|$ is at least $2\beta m_0 n$. Also, since each example is of length 1, it is easy to see that

Claim 1 *With high probability,* $|x_{update} - \mathbf{E}[x_{update}]| \leq \sqrt{m} \log n$.

We now consider the two cases mentioned above.

Case 1: Suppose

$$|x_{update}| \leq \left(\frac{1-2\eta}{1-\eta}\right) m\epsilon\sigma - \sqrt{m}\log n.$$

In this case, we have with high probability that $|\text{Sum}[S_{error}]| \leq m\epsilon\sigma$ using equation (6). But, by definition, each point $x \in S_{error}$ is a unit vector satisfying $-\cos(x, w) > \sigma$, and so $|\text{Sum}[S_{error}]| > \sigma |S_{error}|$. Therefore, $|S_{error}| \leq m\epsilon$ and we have achieved the goal of having a good classifier on $S_{correct} \cup S_{error}$.

Case 2: Suppose

$$|x_{update}| > \left(\frac{1-2\eta}{1-\eta}\right) m\epsilon\sigma - \sqrt{m}\log n.$$

In this case, we know that with high probability,

$$\begin{aligned} \frac{w^* \cdot x_{update}}{|x_{update}|} &\geq \frac{w^* \cdot \mathbf{E}[x_{update}]}{|x_{update}|} - \frac{\sqrt{m}\log n}{|x_{update}|} \\ &\geq \frac{-\sqrt{m}\log n}{|x_{update}|} \end{aligned}$$

which implies (4). To verify (3) we use

$$\begin{aligned} \mathbf{E}(w \cdot x_{update}) &= \left(\frac{1-2\eta}{1-\eta}\right) w \cdot Sum[S_{error}] \\ &\leq -\left(\frac{1-2\eta}{1-\eta}\right) |S_{error}| \sigma |w| \\ &\leq -\sigma |w|/2, \end{aligned}$$

since with high probability

$$|S_{error}| \geq \frac{1-\eta}{2(1-2\eta)}.$$

Therefore the conditions of the Algorithm are satisfied and we have Theorem 1.

We now deal with Assumption A. The simplest idea is to use a new independent set of samples for each iteration. This new set is clearly independent of the current vector w which depends only on previous samples. Assumption A is satisfied at the expense of many more samples than are actually needed.

Alternatively, we know that (6) holds for every *fixed* w and we will argue that with high probability Claim 1 is true *simultaneously* for every w. For example consider

$$\begin{aligned}&|\mathrm{Sum}[S^{\eta}_{error}] - ((1-\eta)\mathrm{Sum}[S_{error}] - \eta\mathrm{Sum}[S_{correct}])| \\ &\quad \le \ |\sum_{w\cdot x < -\sigma} [\theta_+(x) - (1-\eta)]x| + |\sum_{w\cdot x > +\sigma} [\theta_-(x) - \eta]x|, \end{aligned} \tag{7}$$

where $\theta_+(x) + \theta_-(x) = 1$ and $\theta_+(x) = 1$ if x is not corrupted and 0 otherwise. For a fixed w, the sums are unlikely to be very large. Indeed, assuming $|w| = 1$,

$$\Pr(|\sum_{w\cdot x < -\sigma} [\theta_+(x) - (1-\eta)]x| \ge \sqrt{mn}\log n) \le e^{-n(\log n)^2}.$$

Furthermore, in showing all such sums are small with high probability, we need only consider the $\le \binom{|S|}{n} = e^{O(n\log n)}$ half spaces which contain n points of S. Thus the deviation allowed in Claim 1 needs to be increased to $\sqrt{mn}\log n$. This has already been allowed for in the definition of m_0.

The above discussion assumes that η is known to the algorithm. If η is not known, one standard fix is to simply run the algorithm multiple times, each time with a different guessed value in $\{0, 1/|S|, 2/|S|, \ldots, 1/2\}$. However, in our case there is also a less time-consuming fix. Notice that as we increase our guess for η towards $1/2$, the multiple of $\mathrm{Sum}[S_{error}]$ appearing in $\mathbf{E}[x_{update}]$ decreases (but remains positive) and the multiple of $\mathrm{Sum}[S_{correct}]$ increases (and becomes positive once we exceed the true value). In other words, our performance with respect to criteria (3) drops but our performance with respect to (4) improves. But, we can always check if (3) is satisfied. Thus, we may simply choose as large a guess of η as possible that still satisfies (3).

5 Proof of Lemma 1

For a set $S \subseteq \boldsymbol{R}^n$ (S need not be finite) and a distribution μ on R^n, let

$$W_\mu(S) = \{w \in \boldsymbol{R}^n : \ \mathbf{E}_\mu[(w^T x)^2 \mid x \in S] \le 1\}.$$

We will drop the subscript μ (on both W and on the expectation $\mathbf{E}$) when the distribution is clear from context. The key to our proof is the following lemma.

Lemma 2 *Let μ be a measure on $\boldsymbol{R}^n$ which is not concentrated on a subspace of dimension less than n (i.e. the total measure on any subspace of dimension less than n is less than 1). Then, for any $0 < \alpha < 1/3n$, $\beta = 36n^3/\alpha$ and n sufficiently large, there exists an ellipsoid $S \subseteq \boldsymbol{R}^n$ such that*

(a) $\Pr(x \notin S) \le \alpha$.

(b) *Either*

(i) *for all* $w \in \boldsymbol{R}^n$, $\max\{(w^T x)^2 : x \in S\} \leq \beta \mathbf{E}((w^T x)^2 \mid x \in S)$, *or*

(ii) $vol(W(S)) \geq 2 vol(W(\boldsymbol{R}^n))$.

Proof. Let

$$\begin{aligned} M &= \mathbf{E}(xx^T) \\ &= A^2, \end{aligned}$$

where A is symmetric, and non-singular by assumption. Then

$$\mathbf{E}((w^T x)^2) = w^T M w$$

for all $w \in \boldsymbol{R}^n$. Now let

$$\begin{aligned} E &= \{x \in \boldsymbol{R}^n : (w^T x)^2 \leq w^T M w, \forall w \in \boldsymbol{R}^n\} \\ &= \{x \in \boldsymbol{R}^n : ((Aw)^T (A^{-1}x))^2 \leq |Aw|^2, \forall w \in \boldsymbol{R}^n\} \\ &= \{x \in \boldsymbol{R}^n : |A^{-1}x| \leq 1\}. \end{aligned}$$

Note that this shows that E is an ellipsoid. Putting $z = A^{-1}x$ we see that for any $\gamma > 0$,

$$\begin{aligned} \Pr(x \notin \gamma E) &= \Pr(|z| > \gamma) \\ &\leq \sum_{j=1}^{n} \Pr(|z_j| \geq \gamma/\sqrt{n}) \\ &\leq n\gamma^{-2} \sum_{j=1}^{n} \mathbf{E}(z_j^2), \end{aligned}$$

by the Chebychef inequality.

But,

$$\begin{aligned} \mathbf{E}(zz^T) &= \mathbf{E}(A^{-1}xx^T A^{-1}) \\ &= I \end{aligned}$$

and so

$$\Pr(x \notin \gamma E) \leq n^2/\gamma^2.$$

We now take $\gamma = n/\alpha^{1/2}$, $S = \gamma E$ and we see that (a) of the lemma is satisfied.

We now consider two possibilities:

Case (i)

$$\mathbf{E}((w^T x)^2 \mid x \in S) \geq \gamma^2 \mathbf{E}((w^T x)^2)/\beta$$

for all $w \in \boldsymbol{R}^n$.

In this case

$$\begin{aligned} \max\{(w^T x)^2 : x \in S) &\leq \gamma^2 \mathbf{E}((w^T x)^2) \\ &\leq \beta \mathbf{E}((w^T x)^2 \mid x \in S). \end{aligned}$$

Case (ii) There exists $\hat{w} \in \boldsymbol{R}^n$ such that

$$\mathbf{E}((\hat{w}^T x)^2 \mid x \in S) < \gamma^2 \mathbf{E}((\hat{w}^T x)^2)/\beta. \tag{8}$$

Let

$$M_1 = \mathbf{E}(xx^T \mid x \in S).$$

We complete the lemma by showing that

$$\mathrm{vol}(T_1) \geq 2\mathrm{vol}(T), \tag{9}$$

where

$$\begin{aligned} T &= W(\boldsymbol{R}^n) \\ &= \{w \in \boldsymbol{R}^n : \ w^T M w \leq 1\} \end{aligned}$$

and

$$\begin{aligned} T_1 &= W(S) \\ &= \{w \in \boldsymbol{R}^n : \ w^T M_1 w \leq 1\} \end{aligned}$$

It will be convenient to show that

$$\mathrm{vol}(AT_1) \geq 2\mathrm{vol}(AT), \tag{10}$$

which is equivalent to (9) because the linear transformation A multiplies volumes by $|det(A)|$.

Note next that by substituting $v = Aw$ we see that

$$\begin{aligned} AT &= \{v \in \boldsymbol{R}^n : \ v^T A^{-1} M A^{-1} v \leq 1\} \\ &= \{v \in \boldsymbol{R}^n : \ v^T v \leq 1\} \\ &= B_n, \end{aligned}$$

where B_n is the unit ball in $\boldsymbol{R}^n$.

Furthermore,

$$\mathbf{E}((w^T x)^2 \mid x \in S) \leq (1-\alpha)^{-1} \mathbf{E}((w^T x)^2)$$

which follows from $\mathbf{E}((w^T x)^2) \geq \mathbf{E}((w^T x)^2 \mid x \in S) \Pr(x \in S)$. So,

$$\begin{aligned} AT_1 &= \{v \in \boldsymbol{R}^n : \ v^T A^{-1} M_1 A^{-1} v \leq 1\} \\ &\supseteq \{v \in \boldsymbol{R}^n : \ v^T A^{-1} M A^{-1} v \leq 1-\alpha\} \\ &= (1-\alpha) B_n. \end{aligned} \tag{11}$$

Also, AT_1 contains a vector of length $\lambda = \beta^{1/2}/\gamma$. Indeed, let

$$\hat{v} = \lambda \frac{A\hat{w}}{|A\hat{w}|}.$$

Then, from (8),

$$\begin{aligned} \hat{v}^T A^{-1} M_1 A^{-1} \hat{v} &= \frac{\lambda^2}{|A\hat{w}|^2} \hat{w} M_1 \hat{w}^T \\ &\leq \frac{\lambda^2}{|A\hat{w}|^2} \frac{\gamma^2}{\beta} \hat{w} M \hat{w}^T \\ &= 1. \end{aligned}$$

Since AT_1 contains an $n-1$ dimensional ball around the origin and a point at a distance of $1-\alpha$ from the center of the ball, from the convexity of AT_1 it follows then that AT_1 contains a cone with base an $(n-1)$-dimensional ball of radius $1-\alpha$ and height λ.

Thus if V_n denotes the volume of B_n we see that

$$\begin{aligned}\frac{\text{vol}(AT_1)}{\text{vol}(AT)} &\geq \frac{\lambda V_{n-1}(1-\alpha)^{n-1}}{nV_n}\\ &\geq \frac{\lambda(1-\alpha)^{n-1}}{2\sqrt{n}}\\ &\geq 2.\end{aligned}$$

□

We now specialize the above result to the case where μ is concentrated on I_b^n. Let $L_0 = \{x \in I_b^n : \mu(x) \geq 2^{-3nb}\}$. So, $\mu(L_0) \geq 1 - |I_b^n| 2^{-3nb} \geq 1 - 2^{-nb}$.

Let μ_0 denote the measure induced on L_0 by μ i.e. $\mu_0(x) = \mu(x)/\mu(L_0)$ for $x \in L_0$. We consider applying the construction of Lemma 2, K times starting with μ_0. In general we would expect to construct a sequence of ellipsoids S_i This assumes Case (bii) always occurs. Let μ_i denote the measure induced on $S_1 \cap S_2 \cap \cdots \cap S_i$ by μ_0. It is possible that μ_i is concentrated on a subspace V_i of lower dimension. If so, we simply work within V_i from then on. This cannot happen more than n times.

Suppose that Case (bi) never occurs. Then there exists a subspace V_K of dimension ν and ellipsoids $S_1, S_2, \ldots, S_K$ such that if $T_K = L_0 \cap S_1 \cap S_2 \cap \cdots \cap S_K \cap V_K$ then

(a) $\dim(T_K) = \nu$.

(b) $\mu_0(T_K) \geq 1 - \alpha K$.

(c) $\text{vol}_\nu(W(T_K)) \geq 2^{K/n}$,

where in (c),

$$W(T_K) = \{w \in V_K : \mathbf{E}((w^Tx)^2 \mid x \in T_K) \leq 1\}.$$

Part (c) takes into account the doubling of volume K times, and restarting each time we move to a lower dimensional subspace (at most n times).

The above is not possible for sufficiently large K as we will now show. By assumption, T_K contains ν linearly independent vectors $v_1, v_2, \ldots, v_\nu \in I_b^n$. But then

$$\begin{aligned}\mathbf{E}((w^Tx)^2 \mid x \in T_K) &\geq \sum_{i=1}^{\nu}(w^Tv_i)^2\mu_K(v_i)\\ &\geq 2^{-3nb}\sum_{i=1}^{\nu}(w^Tv_i)^2.\end{aligned}$$

So if $w \in W(T_K)$ then

$$\sum_{i=1}^{\nu}(w^Tv_i)^2 \leq 2^{3nb}. \tag{12}$$

Let B denote the $n \times n$ matrix $\sum_{i=1}^{\nu} v_iv_i^T$ so that

$$w^TBw = \sum_{i=1}^{\nu}(w^Tv_i)^2. \tag{13}$$

Let B have eigenvalues $0 = \lambda_1 = \lambda_2 = \cdots = \lambda_{n-\nu} < \bar{\lambda} = \lambda_{n-\nu+1} \leq \lambda_{n-\nu+2} \leq \cdots \lambda_n$. Let $a_1, a_2, \ldots, a_n$ be a corresponding orthonormal basis of eigenvectors. Now if $w = \sum_{i=1}^n u_i a_i$ then $|w|^2 = \sum_{i=1}^n u_i^2$ and $w^T B w = \sum_{i=1}^n \lambda_i u_i^2$ and so

$$\frac{w^T B w}{w^T w} \geq \bar{\lambda} \text{ whenever } w^T B w > 0. \tag{14}$$

But if $w \in V_K$ then $w^T B w > 0$ since $w^T v_i \neq 0$ for at least one i and we can apply (13). But $\bar{\lambda} \neq 0$ is a root of a polynomial of degree at most $n-1$ with rational coefficients α_i/β_i where $|\alpha_i|, |\beta_i| \leq n!2^{nb}$. By a simple computation, this implies that $\bar{\lambda} \geq (n!2^{nb})^{-2n}$ and so (12), (13 and (14) imply that if $w \in W(T_K)$ then

$$|w|^2 \leq 2^{3nb} 2^{2n^2 b} (n!)^{2n} \leq 2^{3n^2 b}$$

(for $b > \log n$) and so

$$\mathrm{vol}_\nu(W(T_K)) \leq (2^{3n^2 b})^{n/2}.$$

This is a contradiction for $K \geq K_0 = \frac{3}{2} n^4 b$. We deduce then that

Theorem 4 *For any $0 < \alpha < 1/3n$ and $\beta = 36n^3/\alpha$ and μ concentrated on I_b^n, there exist $k \leq K_0$ ellipsoids S_i such that if $S = \bigcap_{i=1}^k S_i$*

(i) $\mu(S) \geq 1 - k\alpha - 2^{-nb}$.

(ii) $\max\{(w^T x)^2 : \ x \in S\} \leq \beta \mathbf{E}((w^T x)^2 \mid x \in S)$, *for all $w \in \boldsymbol{R}^n$.*

The previous discussion has been existential in nature and we now show how to make it constructive. This is relatively easy for a finite set of m points (i.e μ is concentrated on the m points). Now if we apply the above theorem to μ then all of the ellipsoids and subspaces are computable in polynomial time.

One way to view the algorithm is the following. We wish to find a set of points with the property that in any direction w, the maximum squared value of the projection of points in that direction is not much more than the average. If initially there is a direction where this is not true, we apply a transformation to the points ($A^{-1}x$, above) that results in their inertial ellipsoid becoming the unit ball. Then we drop all points outside a multiple γ of this ellipsoid and repeat on the smaller set of points (with their original coordinates). This cannot go on forever since we assume that the points are represented by bounded rationals and an associated ellipsoid is doubling in volume at each iteration.

Note that we can make the method constructive for the infinite case as well by picking a sample of points and applying VC-dimension arguments.

6 Open Problems

We list here two open problems related to the topic of this paper. The first is whether it is possible to achieve PAC-learning of linear threshold functions in the presence of random classification noise, using a hypothesis that itself is a linear threshold function (as opposed to a decision list of linear threshold functions as in this paper). In the context of linear programming, one could state this question as follows: suppose one has a feasible set of linear inequalities $Ax > 0$, but then 10% of the rows of A are negated at random to produce the matrix $\tilde{A}$ that is actually presented to the algorithm. Is there an algorithm than with reasonable probability produces a solution x that satisfies nearly 90% of the constraints of $\tilde{A}$? (At least for sufficiently (polynomially) many constraints.)

A second open question is whether weak-learning is possible in the presence of *adversarial* noise. For instance, given a set of examples that are nearly (90%) linearly separable, can one find a linear threshold function that correctly classifies at least a $1/2 + 1/poly(n, b)$ fraction? More generally, one could present this question in somewhat cryptographic terms: given access to $\langle$example, label$\rangle$ pairs drawn from a distribution D, where D satisfies the property than there is some linear threshold function that agrees with D over 90% of the labelings, can one in polynomial time be able to predict the label given to a new example drawn from D with probability at least $1/2 + 1/poly(n, b)$? Known reductions show that a positive answer to this question would imply an $n^{polylog(n)}$-time algorithm for learning DNF formulas, and more generally, AC^0 circuits, over arbitrary distributions [ABFR91].

References

[ABFR91] J. Aspnes, R. Beigel, M. Furst, and S. Rudich. The expressive power of voting polynomials. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, pages 402–409, May 1991.

[AD93] J. A. Aslam and S. E. Decatur. General bounds on statistical query learning and PAC learning with noise via hypothesis boosting. In *Proceedings of the 34th Annual Symposium on Foundations of Computer Science*, pages 282–291, November 1993.

[AD94] J. A. Aslam and S. E. Decatur. Improved noise-tolerant learning and generalized statistical queries. Technical Report TR-17-94, Harvard University, July 1994.

[Agm54] S. Agmon. The relaxation method for linear inequalities. *Canadian Journal of Mathematics*, 6(3):382–392, 1954.

[Ama94] E. Amaldi. *From finding maximum feasible subsystems of linear systems to feed-forward neural network design.* PhD thesis, Swiss Federal Institute of Technology at Lausanne (EPFL), October 1994. (Ph.D. dissertation No. 1282, Department of Mathematics).

[AR88] J. A. Anderson and E. Rosenfeld, editors. *Neurocomputing: Foundations of Research.* MIT Press, 1988.

[Byl93] T. Bylander. Polynomial learnability of linear threshold approximations. In *Proceedings of the Sixth Annual Workshop on Computational Learning Theory*, pages 297–302. ACM Press, New York, NY, 1993.

[Byl94] T. Bylander. Learning linear threshold functions in the presence of classification noise. In *Proceedings of the Seventh Annual Workshop on Computational Learning Theory*, pages 340–347. ACM Press, New York, NY, 1994.

[Fre92] Y. Freund. An improved boosting algorithm and its implications on learning complexity. In *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pages 391–398. ACM Press, 1992.

[Gal90] S. Gallant. Perceptron-based learning algorithms. *IEEE Transactions on Neural Networks*, 1(2):179–191, 1990.

[Kar84] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984.

[Kea93] M. Kearns. Efficient noise-tolerant learning from statistical queries. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, pages 392–401, 1993.

[Kha79] L. G. Khachiyan. A polynomial algorithm in linear programming. *Soviet Mathematics Doklady*, 20:191–194, 1979.

[KV94] M. Kearns and U. Vazirani. *An Introduction to Computational Learning Theory.* MIT Press, 1994.

[MP69] M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry.* The MIT Press, 1969.

[MT89] W. Maass and G. Turán. On the complexity of learning from counterexamples. In *Proceedings of the Thirtieth Annual Symposium on Foundations of Computer Science*, pages 262–267, October 1989.

[Ros62] F. Rosenblatt. *Principles of Neurodynamics.* Spartan Books, 1962.

[Sch90] R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[VC71] V. N. Vapnik and A. Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its applications*, XVI(2):264–280, 1971.