Estimating a Boolean Perceptron from its Average Satisfying Assignment: A Bound on the Precision Required

Paul W. Goldberg*

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

Abstract. A boolean perceptron is a linear threshold function over the discrete boolean domain $\{0,1\}^n$. That is, it maps any binary vector to 0 or 1 depending on whether the vector's components satisfy some linear inequality. In 1961, Chow [9] showed that any boolean perceptron is determined by the average or "center of gravity" of its "true" vectors (those that are mapped to 1). Moreover, this average distinguishes the function from any other boolean function, not just other boolean perceptrons.
We address an associated statistical question of whether an empirical estimate of this average is likely to provide a good approximation to the perceptron. In this paper we show that an estimate that is accurate to within additive error $(\epsilon/n)^{O(\log(1/\epsilon))}$ determines a boolean perceptron that is accurate to within error ϵ (the fraction of misclassified vectors). This provides a mildly super-polynomial bound on the sample complexity of learning boolean perceptrons in the "restricted focus of attention" setting. In the process we also find some interesting geometrical properties of the vertices of the unit hypercube.

1 Introduction

A *boolean perceptron* is a linear threshold function over the domain of 0/1-vectors. (Subsequently we usually just say "perceptron", and omit the adjective "boolean".) Thus it is specifed by a weight vector $\mathbf{w} = (w_1, \ldots, w_n)$ of real numbers and a real-valued threshold t, and it maps a binary vector $\mathbf{x}$ to the output value 1 provided that $\mathbf{w}.\mathbf{x} \geq t$, otherwise it maps $\mathbf{x}$ to 0.

In this paper we consider the problem of estimating a perceptron from an approximate value of the mean, or "center of gravity" of its satisfying assignments. Chow [9] originally showed that the exact value of the average of the satisfying assignments of a boolean perceptron determines it, in that there are no other boolean functions of any kind for which the average satisfying assignment is the same. (Bruck [8] also gives a more general result.) The question of the extent to which an approximation to the average determines the perceptron, is equivalent to a problem of learning boolean perceptrons in the "restricted focus of attention" setting, described below.

* Partially supported by the IST Programme of the EU under contract number IST-1999-14186 (ALCOM-FT).

The *Chow parameters* of a boolean perceptron are the coordinates of the vector sum of the satisfying vectors, together with the number of satisfying vectors. Subject to a uniform distribution over boolean vectors, these are essentially equivalent to the conditional probabilities of the output value 1, conditioned on some input value being set to either 0 or 1. Letting y denote the output value and $\mathbf{x} = (x_1, \ldots, x_n)$, these are the probabilities $Pr(y = 1 \mid x_i = 1)$, for $i = 1, \ldots, n$, together with the value $Pr(y = 1)$.[1] These values determine the functional behavior on binary input vectors, obviously there may some amount of freedom to vary the values of $\mathbf{w}$ and t while preserving the output for any binary input.

In this paper we show that additive approximations of the Chow parameters determine the approximate behavior of the function, to within a mildly super-polynomial factor. That is in contrast to the situation for the weights-based parameterization of a perceptron, for which a tiny perturbation of some parameter may result in a large change to the set of points that are mapped to output value 1. Thus in a sense the Chow parameters are a more robust parameterization.

1.1 Learning-theoretic Background and Results

The related literature can be divided into three parts: some earlier papers that follow on from Chow's paper, work on boolean perceptron learning in general, and work on learning in the so-called "restricted focus of attention" setting.

Earlier work that followed on from [9] includes an algorithm by Kaszerman [14] for recovering a linear threshold function from its Chow parameters. The algorithm is iterative and somewhat related to the perceptron algorithm [17]; it does not have a good bound on the number of iterations and assumes exact values of the parameters are given. A paper of Winder [18] compares 7 functions (4 of which were proposed in previous papers) for rescaling Chow parameters to obtains weights for a linear-threshold function. None of these functions has perfect performance, and it is uncertain that any function exists from individual Chow parameters to good weights – it may be necessary to deal with them collectively rather than individually. A further paper by Winder [19] investigates the class of boolean functions that are uniquely defined by their Chow parameters, and shows among other things that it lies properly between the class of linear threshold functions and the class of monotonic functions.

There is much known about learning boolean perceptrons in various settings, for example irrelevant attributes [15], classification noise [6] and learning from a source of "helpful" examples [2]. Special cases include monomials, decision lists [16, 12] and boolean threshold functions. Further knowledge on this topic occurs in the more general context of perceptrons over the real as opposed to the boolean domain. An example is that they may be PAC-learned in time

[1] If the coordinates of the sum of all satisfying vectors are rescaled down by the number of satisfying vectors, one obtains the average satisfying assignment, whose coordinates are the probabilities $Pr(y = 1 \mid x_i = 1)$. The Chow parameters are recovered by multiplying this average by $2^n \cdot Pr(y = 1)$.

polynomial in the dimension n and the PAC parameters ϵ and δ, using the VC dimension theory [7]. Chapter 24 of [1] and references therein is a good introduction to results on learning boolean perceptrons.

Restricted focus of attention (RFA) learning was introduced and developed in the papers [3–5]. The k-RFA setting (where k is a positive integer) allows the learner to see only a subset of size k of the input attributes of any training example. The usual assumption has been that the input distribution is known to be a product distribution (with no other information given about it). In [13] we studied in detail the problem of learning linear-threshold functions over the real domain in the 1-RFA setting, so that each example of input/output behavior of the target function, has only a single input component value, together with the binary value of the output, revealed to the learner. We showed that the input distribution (in [13], not necessarily a product distribution) needs to be at least partly known, and that the sample-size required for learning depends sensitively on the input distribution. We identified measures of "well-behavedness" of the input distribution and gave sample size bounds in terms of these measures.

This paper addresses the topic of 1-RFA learning of perceptrons where the input distribution is uniform over V, the vertices of the unit hypercube. This learning-theoretic characterization is equivalent because from [5] we have that a random sample of 1-RFA data is equivalent, in terms of the information it conveys, to approximations of the conditional probabilities $Pr(y = 1 \mid x_i = b)$, for $b \in \{0, 1\}$, together with the probability $Pr(y = 1)$, and these approximations have additive error inversely proportional to the sample size. "1-RFA data" means observations of the input/output behavior of the target perceptron in which only one out of the n input values (entry of input vector $\mathbf{x}$) in any training example is made available to the learner, and that entry is chosen by the learner without any other information about that example (equivalently for polynomial learnability, is chosen uniformly at random). The reason why this special case (for which bounds of [13] are inapplicable) is of particular interest is that the uniform distribution over V is the most interesting and widely studied input distribution from the perspective of computational learning theory. The question of whether this learning problem is solvable with polynomial time or sample size is previously discussed in [10] and [13]. Birkendorf et al [5] suggest the following rule: for $1 \leq i \leq n$ and $b \in \{0, 1\}$, let p_b^i be the conditional probability $Pr(y = 1 \mid x_i = b)$ and let $p = Pr(y = 1)$. Then take $\mathbf{x}$ to be a positive instance if $\frac{1}{n} \sum_{i=1}^{n} p_{x_i}^i > p$, otherwise label $\mathbf{x}$ as negative. It is left as an open problem whether the rule is valid.

The computational learning-theoretic result that we obtain is a mildly super-polynomial bound (of the order of $\log(\delta^{-1})(n/\epsilon)^{-\log \epsilon}$) on the sample complexity of learning a perceptron from 1-RFA data. This is a purely "information-theoretic" result; we do not have any algorithm whose guaranteed time complexity improves substantially on a brute-force approach.

1.2 Some Notation and Terminology

Let V be the input domain, *i.e.* the vertices of the unit hypercube, or 0/1-vectors.

Let F and G be two boolean perceptrons, where generally F will denote the target function and G some alternative function, such as a hypothesis returned by an algorithm. The *positive* (resp. *negative*) examples of a function are those that are mapped to 1 (resp. 0). Let $pos(F)$, $neg(F)$, $pos(G)$, $neg(G)$ denote the positive and negative examples of F and G. (So $pos(F) = F^{-1}(1)$, etc.) F and G divide V into 4 subsets defined as follows.

$$V_{00} = neg(F) \cap neg(G) \quad V_{01} = neg(F) \cap pos(G)$$
$$V_{10} = pos(F) \cap neg(G) \quad V_{11} = pos(F) \cap pos(G)$$

For $R \subseteq \mathbf{R}^n$, let $m(R)$ be the number of elements of V that lie in R. Let $a(R)$ be the vector sum of elements of $V \cap R$. Let $\mu(R)$ denote the (unweighted) average of members of V that lie in the region R, so that $\mu(R) = a(R)/m(R)$, well-defined provided that $m(R) > 0$. For any boolean function f, let $\mu(f)$ be the (unweighted) average of its satisfying assignments. The region of disagreement of F and G is $V_{01} \cup V_{10}$; thus the disagreement rate between F and G, over the uniform distribution on V, is $(m(V_{01}) + m(V_{10}))/2^n$.

2 Geometric Results

In this section we give various geometric results about the vertices of the unit hypercube, which we use in section 3 to deduce the sample-complexity bound. We summarize the results of this section:

1. Lemma 1 gives a simple upper bound on the number of elements of V contained in a linear subspace, in terms of the dimension of that subspace.
2. Theorem 1 gives a bound on the bit complexity of the parameters of a hyperplane, in terms of the number of elements of V contained in that hyperplane (the more elements of V contained, the less bit complexity is needed for the parameters).
3. Theorem 2 uses theorem 1 to show that any hyperplane that "narrowly misses" a large fraction of V can be perturbed slightly so that it actually contains all those vertices. The resulting hyperplane does no longer "narrowly miss" any other vertices. More precisely, if a hyperplane comes within distance $2\alpha/n^2$ of a fraction α of the 2^n vertices, then all those $\alpha \cdot 2^n$ vertices lie on the perturbed hyperplane. That result is then used to show limits on the bit complexity of boolean perceptrons that satisfy certain conditions.
4. Theorem 3 derives a lower bound on the distance between $\mu(V_{01})$ and $\mu(V_{10})$ (the means of the two regions of disagreement between a pair of perceptrons) in terms of their disagreement rate $m(V_{01} \cup V_{10})/2^n$.

We generally assume that weight vectors of linear threshold functions are normalized, *i.e.* weights and thresholds are rescaled so that the weight vector is of length 1. By a *vertex* we mean a member of V, *i.e.* a 0/1-vector of length n.

Throughout, when we refer to subspaces, or spanning, or dimension, we mean in the affine sense, so that a "subspace" does not necessarily contain the origin, and a spanning set of S is a set of points in S such that any other point in S is expressible as the sum of one member of the spanning set plus a weighted sum of differences between pairs of points in the spanning set.

Lemma 1. *Any affine subspace S of $\mathbf{R}^n$ of affine dimension d contains at most 2^d elements of V.*

Proof. Proof by induction on d. It clearly holds for $d = 0$, when the subspace consists of a single point.

For $d > 0$ we consider 2 cases. Suppose some subcube of V of dimension d has all its 2^d points in S. Then S cannot contain any other points in V, since then S would have dimension $> d$.

Alternatively, we can divide V into 2 subcubes V_0 and V_1 such that each subcube contains some elements of S and in addition each subcube contains some non-elements of S. S restricted to the span of V_0 (similarly V_1) is a subspace of dimension $d-1$. The inductive hypothesis says that each of these intersections has at most 2^{d-1} points. ♢

The following theorem tells us that when a large fraction of vertices of the unit hypercube span a hyperplane, then the hyperplane must be defined by a linear equation with "simple" coefficients (low bit complexity). The subsequent corollary shows that when a large fraction of vertices span a lower-dimensional subspace, then it can be expressed as the intersection of a set of hyperplanes with simple coefficients. In what follows, logarithms are to the base 2.

Theorem 1. *Let S be a hyperplane of $\mathbf{R}^n$, and suppose that S contains a fraction α of the vertices of the unit hypercube. Assume that the vertices contained by S span S. Suppose that S is the set of points $\{\mathbf{x} : \mathbf{w}.\mathbf{x} = t\}$, where $\mathbf{w} = (w_1, \ldots, w_n)$, $w_1 = 1$ and $0 \leq w_i \leq 1$ for $i = 2, \ldots n$. Then the bit complexity of w_i for $i = 2, \ldots, n$ is at most $\lceil \log(1/\alpha) \rceil$. Hence all the w_i are integer multiples of some constant at least $\alpha/2$.*

Proof. We construct a linear system that must be satisfied by the weights $w_2, \ldots, w_n$, such that when we solve it (invert a matrix) the inverse does not contain numbers of excessive bit complexity.

We have a set $\mathcal{E}$ of $2^n \cdot \alpha$ distinct equations of the form $\mathbf{w}.\mathbf{x} = t$ where $\mathbf{x}$ is a 0/1 vector (a vertex of the hypercube). By our assumption that vertices in S do in fact span S, we have that $\mathcal{E}$ is a linear system that specifies $\mathbf{w}$. Essentially, we want to make a careful choice of just n equations in $\mathcal{E}$ that specify $\mathbf{w}$ and also define a linear system that does not result in numbers with high bit complexity when it is solved. Observe that for two distinct vertices $\mathbf{x}$ and $\mathbf{x}'$ we have $\mathbf{w}.(\mathbf{x} - \mathbf{x}') = 0$, and if we let $\mathbf{w}^- = (w_2, \ldots w_n)$ then (since $w_1 = 1$) we can derive an equation of the form $\mathbf{z}.\mathbf{w}^- = k$ where $\mathbf{z} \in \{0, 1, -1\}^{n-1}$ and k is either 0 or 1. Let $\mathcal{F}$ be the set of all possible equations we can construct in this way.

The linear system we construct will be satisfied by a vector $\mathbf{w}^+$ whose entries $(w_1^+, \ldots, w_{n-1}^+)$ are a permutation of the weights $w_2, \ldots, w_n$, the entries of $\mathbf{w}^-$. The re-ordering of the weights results from the following procedure.

Procedure for Constructing the Linear System:

The following procedure is iterative and we show below that it can continue for at least $n - \lfloor \log(1/\alpha) \rfloor$ iterations. We describe the first three iterations before giving a general description of the k-th iteration.

- Iteration 1: Choose any two equations from $\mathcal{E}$ that differ in the coefficient of some weight w_i, $2 \leq i \leq n$ (which becomes w_1^+, the first weight of $\mathbf{w}^+$), take their difference as described above, and we get an equation (in $\mathcal{F}$) for which the coefficient of w_i (*i.e.* w_1^+) is 1 or -1.
- Iteration 2: Choose any two equations in $\mathcal{E}$ with the same coefficients of w_1^+, identify a weight where they differ, which becomes w_2^+, and their difference gives us a member of $\mathcal{F}$ whose coefficient of w_1^+ is 0 and whose coefficient of w_2^+ is 1 or -1.
- Iteration 3: Choose any two equations in $\mathcal{E}$ with the same coefficients of w_1^+ and w_2^+, identify a weight where they differ, which becomes w_3^+, and their difference gives us a member of $\mathcal{F}$ whose coefficients of w_1^+ and w_2^+ are 0 and whose coefficient of w_3^+ is 1 or -1.
- Iteration k: Choose any two members of $\mathcal{E}$ whose coefficients agree on $w_1^+, \ldots$ w_{k-1}^+, find a weight where they differ, which becomes w_k^+, deduce a member of $\mathcal{F}$ whose first $k-1$ coefficients are 0 and whose k-th coefficient is 1 or -1.

By the pigeon-hole principle, we claim we can continue with this procedure until k exceeds $n - \log(1/\alpha)$. At the k-th iteration there are 2^{k-1} possible vectors of coefficients of $w_1^+, \ldots, w_{k-1}^+$, and $2^n \cdot \alpha$ equations in $\mathcal{E}$ to choose from. Provided that $2^n \cdot \alpha > 2^{k-1}$ we can find a pair of members of $\mathcal{E}$ that agree on these coefficients. Since all members of $\mathcal{E}$ are distinct, they will differ on some other coefficient. The condition $1/\alpha < 2^{n+1-k}$ indicates that for k less than $n + 1 - \log(1/\alpha)$ we can continue.

When we can continue no longer, we supplement our set of $n - \lfloor \log(1/\alpha) \rfloor$ members of $\mathcal{F}$ by any other members of $\mathcal{F}$ that are linearly independent of them. We have constructed a linear system $A.\mathbf{w}^+ = \mathbf{v}$ where

1. A is an invertible $(0/1/-1)$-matrix
2. the submatrix of A that excludes the last $\lfloor \log(1/\alpha) \rfloor$ rows and columns, is upper triangular
3. $\mathbf{v} \in \{0, 1, -1\}^n$, *i.e.* $\mathbf{v}$ is a $(0/1/-1)$-vector.

Hence $\mathbf{w}^+ = A^{-1}\mathbf{v}$. We invert A by Gaussian elimination, identifying a set of linear operations on the rows of A which convert it to the identity matrix I, and then the application of the same operations to I gives A^{-1}. The operations will not create excessively large/small numbers when applied to I.

Procedure for inverting the matrix A

- We start by eliminating the non-zero entries of the upper triangle of the upper triangular submatrix we noted earlier (that is, rows and columns $1, \ldots, n - \lfloor \log(1/\alpha) \rfloor$ of A). This is done by adding or subtracting row $n - \lfloor \log(1/\alpha) \rfloor - 1$ from rows above so as to eliminate non-zero entries in column $n - \lfloor \log(1/\alpha) \rfloor - 1$, then doing the same for row $n - \lfloor \log(1/\alpha) \rfloor - 2$, and so on.
- We then eliminate the region below this submatrix by adding or subtracting copies of the first $n - \lfloor \log(1/\alpha) \rfloor$ rows to/from the bottom rows.
- Then we diagonalize the bottom-right hand submatrix of size $\lceil \log(1/\alpha) \rceil \times \lceil \log(1/\alpha) \rceil$, which may lead to entries of size $1/\alpha$ being created in the diagonal. Re-scale the bottom rows to make the diagonal entries equal to 1.
- Finally we eliminate the region above this bottom-right hand submatrix by adding or subtracting the bottom $\lceil \log(1/\alpha) \rceil$ rows to/from the rows above.

Applying these operations to the identity matrix leads to entries that differ by a factor of at most $2^{\lceil \log(1/\alpha) \rceil}$, hence the bit complexity of entries of A^{-1} is at most $\log(2/\alpha)$. ♢

We use theorem 1 to prove the following result.

Theorem 2. *Let $\alpha \in (0,1)$, and let $\beta \leq 2\alpha/n^2$. Given any affine subspace of $\mathbf{R}^n$ whose β-neighborhood contains a fraction α of points on the unit hypercube, there exists an affine subspace which itself contains all those $2^n \cdot \alpha$ points.*

Proof. Let $V_S \subseteq V$ be the set of vertices within a β-neighborhood of S, where by our assumption, $|V_S| = 2^n \cdot \alpha$. We assume that S is a hyperplane – if it is of lower dimension we can choose some arbitrary hyperplane that contains it.

Let $S = \{\mathbf{x} \ : \ \mathbf{w}.\mathbf{x} = t\}$, where by rescaling we can choose $\|\mathbf{w}\| = 1$. For $\mathbf{x} \in V_S$, we have $\mathbf{w}.\mathbf{x} \in [t - \beta, t + \beta]$.

Define a new weight vector $\mathbf{w}'$ derived from $\mathbf{w}$ by taking each weight in $\mathbf{w}$ and rounding it off to the nearest integer multiple of β. Then products $\{\mathbf{w}'.\mathbf{x} \ : \ \mathbf{x} \in V_S\}$ can take n possible values. To see this, observe that the value of $\mathbf{w}'.\mathbf{x}$ must be an integer multiple of β, and the rounding error (absolute difference $|\mathbf{w}'.\mathbf{x} - \mathbf{w}.\mathbf{x}|$ *i.e.* $|\mathbf{w}'.\mathbf{x} - t|$) is at most $\beta n/2$. Let T be the set of these n values.

Let t' be the member of T which maximizes the number of vertices $\mathbf{x}$ satisfying $\mathbf{w}'.\mathbf{x} = t'$. Then there are at least $2^n \cdot \alpha/n$ vertices $\mathbf{x}$ of the unit hypercube that satisfy $\mathbf{w}'.\mathbf{x} = t'$. We will use this observation to prove that in fact there are no vertices $\mathbf{x} \in V$ for which $\mathbf{w}'.\mathbf{x}$ "narrowly misses" t', in particular this will show that no other element $t'' \in T$ with $t'' \neq t'$ satisfies $\mathbf{w}'.\mathbf{x} = t''$ for any vertex $\mathbf{x}$.

Let S' be a hyperplane containing all points $\mathbf{x}$ that satisfy $\mathbf{w}'.\mathbf{x} = t'$. Suppose that $S' \cap V$ spans the hyperplane S'. (Below we show how to handle the alternative.)

$|S' \cap V| \geq 2^n \cdot \alpha/n$. By theorem 1 we have that for all vertices $\mathbf{v} \in S'$, $\mathbf{w}'.\mathbf{v} - t'$ is an integer multiple of α/n. (Theorem 1 applies to weight vectors $\mathbf{w}$

whose maximum component is 1. They can be normalized by rescaling by a factor of at most n.) Now consider members of V_S that do not lie in S'. These points $\mathbf{v}$ satisfy $\mathbf{w}'.\mathbf{v} - t' \in [-n\beta/2, n\beta/2]$. We chose $\beta \leq 2\alpha/n^2$, and consequently for $\mathbf{v} \in V_S$ to satisfy these two properties, $\mathbf{w}'.\mathbf{v} - t' = 0$. Hence all elements of V_S lie in S.

Now suppose that $S' \cap V$ spans a proper subspace of S'. In this case we embed V in $\mathbf{R}^{n+1}$: let V be the vertices of the unit hypercube of $\mathbf{R}^{n+1}$ for which $x_{n+1} = 0$ and let V' be those for which $x_{n+1} = 1$. $S' \cap V$ spans a subspace of $\mathbf{R}^{n+1}$ of affine dimension d, where $d < n-1$. Choose $n-d$ vertices from V' in general position, and these vertices together with $S' \cap V$ span a hyperplane H of $\mathbf{R}^{n+1}$. Moreover any element of V is at least as close to H as it is to S'. The argument for the first case (where $S' \cap V$ spans S') can now be applied. $\diamondsuit$

The following lemma is used in theorem 3 below.

Lemma 2. *Given boolean perceptrons F and G with V_{01} and V_{10} as defined in section 1.2, suppose we have subspaces $S' \subset S \subseteq \mathbf{R}^n$ with $\dim(S') = dim(S) - 1$ where $S' \cap (V_{01} \cup V_{10})$ contains at least $2^n\alpha$ points, and $\mu(S' \cap V_{01})$ differs from $\mu(S' \cap V_{10})$ by L_2 distance d. Suppose also that S' divides S into 2 connected components such that on each side, F gives the same label to points on that side. Then $\mu(S \cap V_{01})$ differs from $\mu(S \cap V_{10})$ by L_2 distance $d(\alpha/2\sqrt{n})$.*

Proof. We have assumed a gap between the means of points in V_{01} and V_{10} that also belong to S', and we will upper-bound the extent to which points in V_{01} and V_{10} that also belong to $S \setminus S'$ can bring the overall averages together.

The general observation is that if the points in $(S \setminus S') \cap (V_{01} \cup V_{10})$ cause the overall means to come closer together, they do in the process displace those means away from S', due to the fact that points in $(S \setminus S') \cap (V_{01} \cup V_{10})$ must lie within distance $\sqrt{(n)}$ of any point in V, but also are at least a distance $\alpha/2$ from S', due to theorem 1. (This also uses the assumption that one side of $S \setminus S'$ contains only points from V_{01} but not V_{10} and the other contains points from V_{10} but not V_{01}.) $\diamondsuit$

Theorem 3. *Let F and G be boolean perceptrons that disagree on a fraction ϵ of the 2^n members of V. Then the Euclidean distance between $\mu(V_{01})$ and $\mu(V_{10})$ is at least $(\epsilon/n)^{O(\log(1/\epsilon))}$.*

Proof. By a *line* we mean a 1-dimensional affine subspace. If l is a line and S is a set of points, let $l(S)$ denote the set of points obtained by projecting elements of S onto their closest points on l.

Let l be a line normal to P_F, a hyperplane defining F.

Observe that members of $l(V_{01})$ are separated from members of $l(V_{10})$ by the point of intersection of l and P_F (which itself is $l(P_F)$).

Two cases:

Case 1: Suppose at least a fraction ϵ of the members of $V_{01} \cup V_{10}$ (*i.e.* at least $2^n\epsilon^2$ points altogether) have projections onto l that are more than ϵ/n^2 distant from $l(P_F)$. In this case we have

$$|\mu(V_{01}) - \mu(V_{10})| \geq \epsilon^2/n^2.$$

Case 2: in which more than a fraction $(1-\epsilon)$ of points in $V_{01} \cup V_{10}$ lie within distance ϵ/n^2 of $l(P_F)$.

In this case we apply theorem 2 to obtain a hyperplane P' that contains all but a fraction ϵ of points in $V_{01} \cup V_{10}$.

Choose an affine subspace of P' that separates members of V_{01} lying in P' from members of V_{10} lying in P'. Let $l' \subseteq P'$ be a line normal to that subspace.

Two cases (similar to before):

Case 2a: At least a fraction ϵ of the members of $(V_{01} \cup V_{10}) \cap P'$ have projections onto l' that are more than ϵ/n^2 distant from the boundary $l'(P')$. Then the distance between the means of points in P' is at least ϵ^2/n^2. All points in $V_{01} \cup V_{10}$ *not* in P' lie at least ϵ from P' by theorem 1 (coefficients of P' have a common factor approximately ϵ). We claim that the overall distance between $\mu(V_{01})$ and $\mu(V_{10})$ is at least $(\epsilon^2/n^2)(\epsilon(1-\epsilon)/2\sqrt{n})$. This claim follows using lemma 2.

Case 2b: Alternatively, all but a fraction ϵ of these points lie in some subspace of $P'' \subset P'$. We look in that subspace for a similar difference between means of members of V_{01} and members of V_{10} in P''. As before let $l'' \subseteq P''$ be a line constructed in a similar way to l'. Either we find a fraction ϵ of points $\mathbf{x}$ with $l''(\mathbf{x})$ more than ϵ/n^2 from the boundary between $V_{01} \cap P''$ and $V_{10} \cap P''$ or we continue by looking in a subspace of P'' that now contains at least a fraction $(1-\epsilon)^2$ of points in $V_{01} \cap V_{10}$. By lemma 1 this process only continues for $\log(\epsilon^{-1})$ iterations before subspaces do not have sufficient dimension to hold a fraction ϵ of elements of V. $\diamondsuit$

3 Computational Learning-theoretic Consequences

Recall that 1-RFA data refers to the special case of RFA learning where each training example has just a single input attribute revealed to the learner, together with the (binary) output value. In the case of boolean functions, a training example is essentially a member of $\{1, \ldots n\} \times \{0,1\} \times \{0,1\}$, the identity of one of the n attributes, together with its binary value, together with the binary value of the output. The identity of the observed attribute is assumed to be chosen by the learner, which as noted in [13] is equivalent (for the purpose of polynomial bounds) to the assumption that it is chosen at random.

We continue by using the preceding results to obtain a bound on learning boolean perceptrons in the 1-RFA setting. We first show how to determine (using 1-RFA data) which of any given pair of candidate hypotheses F and G that have a disagreement rate of ϵ, is the target function, using time and sample complexity $\log \delta (n/\epsilon)^{-\log \epsilon}$, where δ is the allowed probability that the learner's choice is incorrect. We then show how a computationally unbounded learner can select a good hypothesis from the entire set of boolean perceptrons, using sample size $\log \delta (n/\epsilon)^{-\log \epsilon}$, where δ is the probability that the hypothesis has error $> \epsilon$.

3.1 Distinguishing between two candidate functions

Remark 1. Given any perceptron F we may estimate $Pr(F(\mathbf{x}) = 1)$ and $Pr(F(\mathbf{x}) = 1 \mid x_i = 1)$ within additive error ϵ and with uncertainty δ, in time polynomial in ϵ^{-1} and δ^{-1}.

Proof. This can be done by generating $poly(n, \epsilon^{-1}, \delta^{-1})$ elements $\mathbf{x}$ of V uniformly at random and taking empirical estimates of the relevant probabilities. $\diamondsuit$

It is worth noting that computing these quantities exactly is $\sharp P$-hard as it is the 0/1 knapsack problem [11]. The following fact is noted in [5], with regard to the problem of PAC-learning boolean perceptrons:

Remark 2. Learning from 1-RFA data with respect to the uniform distribution is equivalent to reconstructing a boolean function from empirical estimates of the components of its average satisfying assignment.

These two facts together with the geometric results of the previous section are used to solve the following learning problem: Given two perceptrons F and G, one of which is the target function, and which have a disagreement rate of ϵ, decide which of the two is the target function.

There are two cases that we consider: either the difference between $Pr(V_{01})$ and $Pr(V_{10})$ exceeds ϵ^2/n, or not. In the first case, assume that the difference is more than ϵ^2/n, and consequently $Pr(pos(F))$ and $Pr(pos(G))$ also differ by ϵ^2/n. A simple Chernoff bound analysis shows that a sample size of $O(\log(\delta^{-1}) \cdot \epsilon^2/n)$ is sufficient to distinguish F from G using the relative frequency of positive examples.

Now in the second case, we derive a lower bound on the distance between the means $\mu(pos(F))$ and $\mu(pos(G))$ (using the lower bound on the distance between $\mu(V_{01})$ and $\mu(V_{01})$ in conjunction with the upper bound of ϵ^2/n on the difference between $Pr(V_{10})$ and $Pr(V_{01})$.

We show that the distance between $\mu(V_{01})$ and $\mu(V_{10})$ is at least $\epsilon^{-1} \cdot (\epsilon/n)^{-\log \epsilon}$. This allows us to deduce the distance of $(\epsilon/n)^{-\log \epsilon}$ between $\mu(pos(F))$ and $\mu(pos(G))$ as follows.

$$\mu(pos(F)) = \lambda \cdot \mu(V_{11}) + (1 - \lambda)\mu(V_{10})$$

where $\lambda \geq \epsilon/2$. Similarly

$$\mu(pos(G)) = \lambda' \cdot \mu(V_{11}) + (1 - \lambda')\mu(V_{01})$$

where $\lambda' \geq \epsilon/2$. Hence

$$\mu(pos(F)) - \mu(pos(G)) = \lambda(\mu(V_{01}) - \mu(V_{01})) + (\lambda - \lambda')(\mu(V_{01}) - \mu(V_{11}).$$

By the assumption (of this second case) that V_{01} and V_{10} differ in size by at most a fraction ϵ^2/n elements, we have that $|\lambda - \lambda'| \leq \epsilon^2$. Hence

$$\mu(pos(F)) - \mu(pos(G)) \geq \epsilon(\mu(V_{01}) - \mu(V_{01})).$$

We now use theorem 3 to say that in this case, there is a difference between the means of V_{01} and V_{10} at least $(\epsilon/n)^{O(\log(1/\epsilon))}$, and hence a difference between the means of positive examples of the order of $(\epsilon/n)^{O(\log(1/\epsilon))}$, so that (in conjunction with remark 2) a sample size of $\log(\delta^{-1}) \cdot (n/\epsilon)^{O(\log(1/\epsilon))}$ is sufficient to identify (with probability $1-\delta$ of correctness) which of the two candidate means is the correct one.

3.2 Estimating a target function from the set of all possible perceptrons

We have shown how to identify which of two given perceptrons is correct, using time and sample size of the order of $\log \delta(n/\epsilon)^{O(\log(1/\epsilon))}$. A computationally unbounded learner may identify (for target perceptron F) a hypothesis G as follows.

Obtain empirical estimates of the Chow parameters of F using a sample size of $O(\log \delta(n/\epsilon)^{\log(1/\epsilon)})$. Evaluate the Chow parameters of every boolean perceptron, and output any perceptron G whose Chow parameters are all within $(n/\epsilon)^{\log(1/\epsilon)}$ of the observed parameters.

With probability $1-\delta$, the observed Chow parameters differ from the correct Chow parameters of F by no more than the above margin of $(n/\epsilon)^{\log(1/\epsilon)}$. We have also shown that any G with a disagreement rate of $\geq \epsilon$ with F, must have Chow parameters that differ by more than this margin, so that any hypothesis that is eligible to be chosen according to our rule, does in fact have error less than ϵ.

3.3 Conclusions and Open Problems

The problem of PAC-learning a boolean perceptron from empirical estimates of its Chow parameters has been raised in various papers in computational learning theory. We have so far just shown a bound on sample complexity only (the problem of how to best select the right hypothesis given sufficient data having not been addressed), and that bound is still super-polynomial. The next step would appear to be to find a bound on the sample complexity that is polynomial in the parameters n, ϵ^{-1} and δ^{-1}. The geometrical results we have found may well assist with further progress.

References

1. M. Anthony and P.L. Bartlett (1999). *Neural Network Learning: Theoretical Foundations*, Cambridge University Press, Cambridge.
2. M. Anthony, G. Brightwell and J. Shawe-Taylor (1995). On Specifying Boolean Functions by Labelled Examples. *Discrete Applied Mathematics* **61**, pp. 1-25.
3. S. Ben-David and E. Dichterman (1998). Learning with Restricted Focus of Attention, *J. of Computer and System Sciences*, **56**(3), pp. 277-298. (earlier version in COLT'93)

4. S. Ben-David and E. Dichterman (1994). Learnability with restricted focus of attention guarantees noise-tolerance, *5th International Workshop on Algorithmic Learning Theory*, pp. 248-259.
5. A. Birkendorf, E. Dichterman, J. Jackson, N. Klasner and H.U. Simon (1998). On restricted-focus-of-attention learnability of Boolean functions, *Machine Learning*, **30**, 89-123. (earlier version in COLT'96)
6. A. Blum, A. Frieze, R. Kannan and S. Vempala (1998). A Polynomial-time Algorithm for Learning Noisy Linear Threshold Functions. *Algorithmica* **22**: pp. 35-52.
7. A. Blumer, A. Ehrenfeucht, D. Haussler and M.K. Warmuth (1989). Learnability and the Vapnik-Chervonenkis Dimension, J.ACM **36**, 929-965.
8. J. Bruck (1990). Harmonic analysis of polynomial threshold functions. *SIAM Journal of Discrete Mathematics*, **3** (2), 168-177.
9. C.K. Chow (1961). On the characterization of threshold functions. *Proc. Symp. on Switching Circuit Theory and Logical Design*, 34-38.
10. E. Dichterman (1998). Learning with Limited Visibility. *CDAM Research Reports Series, LSE-CDAM-98-01* 44pp.
11. M. E. Dyer, A. M. Frieze, R. Kannan, A. Kapoor, L. Perkovic and U. Vazirani (1993). A mildly exponential time algorithm for approximating the number of solutions to a multidimensional knapsack problem. *Combinatorics, Probability and Computing* **2**, 271-284.
12. T. Eiter, T. Ibaraki and K. Makino (1998). Decision Lists and Related Boolean Functions. *Institut Für Informatik JLU Giessen (IFIG) Research Reports 9804.*
13. P.W. Goldberg (1999). Learning Fixed-dimension Linear Thresholds from Fragmented Data. *Warwick CS dept. tech. report RR362*, Sept. 99, accepted for publication in *Information and Computation* as of Dec. 2000. A preliminary version is in Procs of the 1999 Conference on Computational Learning Theory, pp. 88-99 July 1999.
14. P. Kaszerman (1963). A geometric test-synthesis procedure for a threshold device. *Information and Control* **6**(4), 381-398.
15. N. Littlestone (1988). Learning Quickly When Irrelevant Attributes Abound: A New Linear-threshold Algorithm. *Machine Learning* **2**, pp. 285-318.
16. R. L. Rivest (1996). Learning Decision Lists. *Machine Learning* **2** pp. 229–246.
17. F. Rosenblatt (1962). *Principles of Neurodynamics.* Spartan Books, New York, 1962.
18. R.O. Winder (1969). Threshold Gate Approximations Based on Chow Parameters. *IEEE Transactions on Computers* **18**, pp. 372-5.
19. R.O. Winder (1971). Chow Parameters in Threshold Logic. *Journal of the ACM* **18**(2), pp. 265-89.