MARTIN MARIETTA ENERGY SYSTEMS LIBRARIES

3 4456 0353255 9

ORNL/TM-11905
(CESAR-91/24)

ornl

OAK RIDGE
NATIONAL
LABORATORY

MARTIN MARIETTA

N-Learners Problem: Learning Boolean Combinations of Halfspaces

Nageswara S. V. Rao
E. M. Oblow
Charles W. Glover

OAK RIDGE NATIONAL LABORATORY
CENTRAL RESEARCH LIBRARY
CIRCULATION SECTION
4500N ROOM 175
LIBRARY LOAN COPY
DO NOT TRANSFER TO ANOTHER PERSON
If you wish someone else to see this report, send in name with report and the library will arrange a loan.

MANAGED BY
MARTIN MARIETTA ENERGY SYSTEMS, INC.
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from the Office of Scientific and Technical Information, P.O. Box 62, Oak Ridge, TN 37831; prices available from (615) 576-8401, FTS 626-8401.

Available to the public from the National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Rd., Springfield, VA 22161.

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

ORNL/TM-11905
CESAR-91/24

Engineering Physics and Mathematics Division

N-LEARNERS PROBLEM: LEARNING BOOLEAN COMBINATIONS OF HALFSPACES

Nageswara S.V. Rao
Department of Computer Science
Old Dominion University
Norfolk, VA 23529-0162

E. M. Oblow
Charles W. Glover

DATE PUBLISHED — March 1992

Research sponsored by the
Office of Engineering Research Program
Basic Energy Sciences
U.S. Department of Energy
and
Office of Naval Research
Intelligent Systems Program
U.S. Department of Defense

Prepared by the
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831
managed by
MARTIN MARIETTA ENERGY SYSTEMS, INC.
for the
U.S. DEPARTMENT OF ENERGY
under contract DE-AC05-84OR21400

MARTIN MARIETTA ENERGY SYSTEMS LIBRARIES
3 4456 0353255 9

CONTENTS

ACKNOWLEDGMENTS v
ABSTRACT vii
1. INTRODUCTION 1
2. LEARNABILITY PROBLEM 3
3. OFF-LINE VERSION 5
3.1 LEARNING A SINGLE SEPARATING PLANE 5
3.2 GENERAL CASE 6
4. ON-LINE VERSION 8
4.1 MULTI-PERCEPTRON THEOREM 8
5. CONCLUSIONS 10
REFERENCES 11

ACKNOWLEDGMENTS

The authors gratefully acknowledge the continuing financial support of this learning research by Oscar Manley of the Basic Energy Sciences Program in the Department of Energy and Teresa McMullen in the Intelligent Systems Program of the Office of Naval Research in the Department of Defense. In addition, N.S.V. Rao is partially funded by National Science Foundation under grant #IRI-9108610, Old Dominion University Summer Faculty Fellowship for 1991 and Virginia's Center for Innovative Technology under contract # INF-90-015.

ABSTRACT

Given two subsets S_1 and S_2 (not necessarily finite) of $\Re^d$ separable by a Boolean combination of N halfspaces, we consider the problem of *learning* (in the sense of Valiant [23]) the separation function from a finite set of examples, i.e. we produce with high probability a function close to the actual separating function. Our solution consists of a system of N perceptrons and a single consolidator which combines the outputs of the individual perceptrons. We show that an *off-line* version of this problem, where the examples are given in a batch, can be solved in time polynomial in the number of examples. We also provide an *on-line* learning algorithm that incrementally solves the problem by suitably training a system of N perceptrons much in the spirit of the classical perceptron learning algorithm. This solution constitutes an example of a composite system of N learners capable of accomplishing a task that is not achievable by a single learner, for a single perceptron is incapable of separating sets that are not linearly separable.

Keywords and Phrases: computation learnability, N-learners problem, N-polyhedral separability, perceptron.

1. INTRODUCTION

Two sets $S_1 \subseteq \Re^d$ and $S_2 \subseteq \Re^d$ (not necessarily finite) are said to be *N-polyhedral separable* if there exists a Boolean combination of N half-spaces of $\Re^d$ that separates S_1 and S_2 [14]. More formally, S_1 and S_2 are *N-polyhedral separable* if there exist N hyperplanes $H_j = \{z : (x^j)^T z = x_0^j\}$, (for $z, x^j \in \Re^d$, $x_0^j \in \Re$, $j = 1, 2, \ldots, N$) that separate S_1 and S_2 through a Boolean formula as follows. Associate with each hyperplane H_j a Boolean variable ξ_j: for any $z \in \Re^d$ the variable ξ_j is true if $(x^j)^T z > x_0^j$ and false if $(x^j)^T z < x_0^j$. A truth value is not defined for points lying on the hyperplanes. A Boolean formula $\psi = \psi(\xi_1, \xi_2, \ldots, \xi_N)$ separates the sets S_1 and S_2 if ψ is true at every point of S_1 and false at every point of S_2. Thus, ψ defines a map $f_\psi : R^d \mapsto \{0, 1\}$, called the *separating function* such that for $x, y \in \Re^d$

(a) $f_\psi(x) = f_\psi(y)$ if both x and y belong to either S_1 or S_2, and

(b) $f_\psi(x) \neq f_\psi(y)$ if one of x and y belongs to S_1 and the other belongs to S_2.

In this paper, we consider the problem of learning an "approximation" to $f_\psi()$ using a finite number of points from $S_1 \cup S_2$ produced according to an unknown probability distribution $P_{S_1 \cup S_2}$. Also, in the case where S_1 and S_2 are finite, we are interested in an incremental algorithm along the lines of the classical perceptron learning algorithm [16]. Meggido [15] considers the cases where S_1 and S_2 are finite; it is shown that for a constant N, the separating function ψ can be computed in time polynomial in $|S_1| + |S_2|$, but, this problem is NP-hard for arbitrary values of N. The popular case of *linear-separability* of S_1 and S_2 (i.e. S_1 and S_2 are separable by a single hyperplane) is a special case of N-polyhedral separability (when $N = 1$). In this case, if additionally S_1 and S_2 are finite, the separating hyperplane can be incrementally learned by the well-known *classical perceptron fixed increment learning procedure* ([16], [18], and [22]). Also, the separating plane can be computed in a batch mode in time polynomial in $|S_1| + |S_2|$ by using linear programming methods [13]. Several variants of this problem can be solved by posing them as quadratic programming problems as shown by Vapnik [24].

For any set $A \subseteq \Re^d$, we define an *indicator function*, $I_A : \Re^d \mapsto \{0, 1\}$ such that for $x \in \Re^d$, we have: $I_A(x) = 1$ if $x \in A$ and $I_A(x) = 0$ if $x \notin A$. A pair of the form $< x, i >$, for $x \in \Re^d$, and $I_{S_i}(x) = 1$ for some $i = 1, 2$ is called an *example* and a sequence of l examples is called an *l-sample*. We consider the problem of *learning* the separation function f_ψ from an l-sample of $S_1 \cup S_2$, produced according to $P_{S_1 \cup S_2}$ such that each example is independent. We consider two formulations of this problem: *on-line* and *off-line*. In the former, the examples are produced in a sequence and we are required to update the separation function (if needed) after every example. In spirit, we are interested in a procedure analogous to the classical perceptron learning procedure [16]. Similar to the case of classical perceptrons, we assume that S_1 and S_2 are both finite in the on-line case. In the *off-line* formulation, we are interested in learning the separation function, in sense of Valiant [23] which has received considerable attention in recent years (see Blumer et al [6], Haussler [10], Littlestone [14], Natarajan [17] for some of works on this topic). Under this

paradigm of *computational learnability*, given a sufficiently large sample, we are required to produce a close approximation to f_ψ with high probability, irrespective of the probability distribution $P_{S_1 \cup S_2}$ used to produce the sample.

Our solutions to both the formulations consist of "suitably" employing a system of N perceptrons and a single consolidator. For the on-line version, we show a convergence theorem which is a generalization of the classical perceptron convergence theorem; each learning step (in response to an example) will take time polynomial in the size of $S_1 \cup S_2$. Note that the conventional perceptron can only separate two sets that are linearly-separable [16], but incapable of separating sets that are N-polyhedral separable. Here our contribution is two-fold: (a) an algorithm to separate classes inseparable by a single perceptron, and (b) a generalization of the conventional perceptron learning algorithm to two-layer perceptrons for this particular case.

Coming to the machine learning aspects, we know of no off-line method that learns to separate S_1 and S_2 by a Boolean combination of N half-spaces; in our case S_1 and S_2 need not be finite (hence we cannot use the algorithm of Meggido [15]). The closest work to the present problem is that of learning a union of half spaces by Baum [3]; this method hinges on "cornering" the given examples by using a suitable convex hull, and it is unclear if this method can be directly applied to the problem of "separating" sets. We show that we can solve the off-line problem by using N perceptrons and a Boolean combination function on the output of the perceptrons. This work follows the spirit of the *N-learners problem* that deals with combining the outputs of N individual learners to produce results "better" than those of any individual learner [20].

The organization of this paper is as follows: We show the basic learnability result in section 2. The off-line version is solved in Section 3, and the on-line version is solved in Section 4.

2. LEARNABILITY PROBLEM

Let X be a set and $C \subseteq 2^X$ denote a *concept class*. Then $H \subseteq 2^Y$, for $Y \subseteq X$, is called a *hypotheses class*. Let P_X be a probability distribution defined on X. We restrict our attention to the case of $X \subseteq \Re^d$. We say that the concept class C is *learnable* by H if for every P_X and any concept $f \in C$, there is some positive integer $l < \infty$ such that given a sample $<x_1, I_f(x_1)>, <x_2, I_f(x_2)>, \ldots, <x_l, I_f(x_l)>$ and ϵ and δ, $0 < \epsilon, \delta < 1$, an *approximation* $h \in H$ can be produced such that

$$Prob[\mu(f \Delta h) \leq \epsilon] \geq 1 - \delta$$

where

$$f \Delta h = (f - h) \cup (h - f)$$

and

$$\mu(f \Delta h) = \int_{I_f(x) \neq I_g(x)} dP_X(x) \quad .$$

Note that $\mu(f \Delta h)$ gives the probability that a randomly chosen $x \in X$ precisely belongs to f or h but not to both.
The above condition is often also expressed as:

$$Prob[\mu(f \Delta h) > \epsilon] \leq \delta \quad .$$

We refer to any of the above two conditions as the (ϵ, δ)-*condition*. We say that C is *polynomially learnable* by H if the number of examples needed to ensure the (ϵ, δ)-condition is a polynomial in $1/\epsilon$ and $1/\delta$. Several variants of this basic problem have been studied by a number of researchers (see [1-4,10,14,17,23,24] for some important works).

One of the important considerations for ensuring learnability is to make H satisfy certain conditions. A family $H \subseteq 2^X$ *shatters* a set $X_l = \{x_1, x_2, \ldots, x_l\} \subseteq X$, if $\{h \cap X_l | h \in H\} = 2^{X_l}$, i.e. for every subset of X_l there exists $h \in H$ that contains this subset but not its complement. The *Vapnik and Chervonenkis dimension* of H, denoted by $VC(H)$ is the maximum $|X_l|$ such that every subset of X_l is shattered by H. The $VC()$ plays a very critical role in learnability in that if $C \subseteq H$, C is learnable if and only if $VC(H) < \infty$ for several important formulations (see [6] for details).

Let the hypothesis set H denote the set of possible Boolean combinations of at most N halfspaces. The VC-dimension of the set of all hyperplanes in $\Re^d$ is $d + 1$ [6]. Let A be a concept class with a finite VC-dimension. It is well known that the VC dimension of the set of concepts obtained by taking the Boolean combinations N elements from A will also be finite [8].

We derive the VC-dimension of H based on a formula for the number of k-dimensional faces (k-faces) of an arrangement formed by N hyperplanes in $\Re^d$ in Edelsbrunner [9] (see also [7]).

Lemma 1. *The VC-dimension of the set of all Boolean combinations of N hyperplanes of dimensionality d is $\sum_{i=0}^{d} \binom{N}{i}$.*

Proof: The number of k-faces formed by N hyperplanes in $\Re^d$ is found in [9] to be

$$f_k^d(N) = \sum_{i=0}^{k} \binom{d-i}{k-i} \binom{N}{d-i} \quad .$$

Imagine the d-faces, i.e. d-dimensional cells, formed by the hyperplanes; each cell corresponds to a unique labeling by the hyperplanes, i.e. the Boolean vector $(\xi_1, \xi_2, \ldots, \xi_N)$ will yield the same value for each point in a cell. A set of points placed one to each cell will be shattered by a suitable Boolean function on the cells. Thus the maximum size of the set that can be shattered by H is $f_d^d(N) = \sum_{i=0}^{d} \binom{N}{d-i} = \sum_{i=0}^{d} \binom{N}{i}$. QED

It has been shown that, if $C = H$, any hypothesis $h \in H \subseteq 2^{\Re^d}$ that is consistent with a sample of sufficiently large size, will be a close predictor of f with a high probability [6]. It has also been shown that given $\max\left(\frac{4}{\epsilon}\log\frac{2}{\delta}, \frac{8}{\epsilon}d\log\frac{13}{\epsilon}\right)$ examples, any consistent hypothesis $h \in H$ will satisfy the (ϵ, δ)-condition, where d is the VC-dimension of C [6]. Using Lemma 1 and Theorem 2.1 of [6], we deduce the following theorem which establishes the existence of a learning algorithm for H.

Theorem 1. *Given $\max\left(\frac{4}{\epsilon}\log\frac{2}{\delta}, \frac{8}{\epsilon}\sum_{i=0}^{d}\binom{N}{i}\log\frac{13}{\epsilon}\right)$ examples of $f \in C$, any $h \in H = C$ consistent with the examples will satisfy $Prob[\mu(h\Delta f) > \epsilon] < \delta$ for any given ϵ and δ, for $0 < \epsilon, \delta < 1$.*

Based on the above theorem and the results of Blumer et al [6], which ensure the (ϵ, δ)-condition is satisfied, it suffices to pick any $h \in H = C$ that correctly classifies all the examples of the given sample. Now the computational feasibility of this method depends on how fast a consistent hypothesis can be computed. In the next section, we address this issue.

3. OFF-LINE VERSION

We first discuss the special case of learning a single separating plane; we will bring together several existing results on this problem to set up for the discussion of the general case. In this section, for ease of presentation we let S_1 and S_2 denote finite sets of examples produced according to the underlying distribution.

3.1 LEARNING A SINGLE SEPARATING PLANE

We first discuss two simple cases: (a) S_1 and S_2 are linearly separable; and (b) H is a set of all half-spaces and S_1 and S_2 are not linearly-separable.

For case (a), since the VC-dimension of the set of all hyperplanes in $\Re^d$ is $d+1$ [6], given $\max\left(\frac{4}{\epsilon}\log\frac{2}{\delta}, \frac{8}{\epsilon}(d+1)\log\frac{13}{\epsilon}\right)$ examples, we can produce a hyperplane that separates the given set of examples which ensures the (ϵ, δ)-condition (Blumer et al [6]). The time complexity of producing such a hypothesis is polynomial in the number of examples. This is shown by Megiddo [15], who reduces this problem to that of linear programming which can be solved in the polynomial time using the algorithms of Karmarkar [12] or Khachiyan [13]. More precisely, let h be the required separating hyperplane; each example defines a linear inequality, i.e., for $x \in S_1$ and $y \in S_2$, we have $x^T h > 0$ and $y^T h < 0$ by augmenting h with an additional last component. This problem can then be solved in polynomial time using the ellipsoid algorithm [13] (see [25] for more general applications of this algorithm).

We can obtain another, rather obvious, algorithm to compute the separating plane of S_1 and S_2 as follows:

(1) Compute the convex hulls of S_1 and S_2, denoted by $Conv(S_1)$ and $Conv(S_2)$ respectively;

(2) Compute the closest pair (p, h) such that $p \in Conv(S_1) \cup Conv(S_2)$ and h is a d-plane of the convex hull to which p does not belong,

(3) Compute a hyperplane parallel to h and halfway between p and h.

Note that the hyperplane h in step (2) supports the corresponding convex hull on one side, and contains on the other side the second convex hull; thus, the hyperplane computed in step (3) can be imagined to be obtained by translating h towards p (in a direction normal to h). Let $|S_1| = n_1$ and $|S_2| = n_2$ and $n = n_1 + n_2$. The complexity of step (1) is $O(n \log n + n^{\lceil (d+1)/2 \rceil})$ in time and $O(n^{\lceil d/2 \rceil})$ in space using the algorithm of Edelsbrunner [9] (see also [11]). Since each convex hull has $O(n^{\lfloor d/2 \rfloor + 1})$ d-faces and $O(n)$ points, step (2) can be carried out using a brute force method in $O(n^{\lceil (d+1)/2 \rceil})$ time (this step can be solved in lesser time using a variant of the algorithm of Bentley and Shamos [5]). The third step can be solved in $O(d)$ time. Thus we have the following theorem.

Theorem 2. *The separating plane of two linearly separable sets with a total of n points in $\Re^d$ can be computed in $O(n \log n + n^{\lceil (d+1)/2 \rceil})$ time and $O(n^{\lceil d/2 \rceil})$ space.*

Moving to case (b), we find that a solution cannot be found that directly. In this case there will not, in general, be a hypothesis in H that is consistent with all the

examples. We, therefore, need to use the more general formulation of Vapnik [24], which requires us to choose a hypothesis with the minimum number of misclassified examples. We then need $\max\left(\frac{8}{\epsilon}\ln\frac{8}{\delta}, \frac{16}{\epsilon}(d+1)\ln\frac{16}{\epsilon}\right)$ examples to ensure the (ϵ, δ) bound. See Appendix A3 of Blumer [6] for a detailed derivation of this bound. Now the computational feasibility of this method depends on our ability to compute the separating hyperplane that minimizes the number of errors.

We now present a method originally proposed by Vapnik [24] (Addendum I) whose complexity was not explicitly stated. Several problems dealing with computing hyperplanes that separate two sets were proposed in [24]. All these problems are reduced to the quadratic programming problem. The general form of the quadratic programming problem deals with minimizing x^TQx, subject to constraints $Ax \leq b$, where $x \in \Re^d$, b is an integer vector, and Q and A are integer matrices. This problem is NP-hard in general but can be solved in polynomial time in the special case where Q is positive definite [19, Problem 16, p. 379]; the polynomial time algorithm is based on the ellipsoid method for linear programming. This problem is reduced to at most n instances of a quadratic programming problem [24], and thus admits a polynomial time solution.

3.2 GENERAL CASE

Now consider the case where S_1 and S_2 are N-polyhedral separable, and H is the set of Boolean combinations of N half-spaces. Based on Theorem 1, we need to ensure that the required hypothesis can be computed in polynomial time. Earlier Megiddo [15] presented a polynomial time algorithm for this problem; this algorithm runs in at least $O(n^{N(d+1)})$ time. We present an alterative algorithm that runs in $O(n^{\lceil (d+1)/2 \rceil + 2})$ time.

We now present propositions shown by Megiddo [15].

Proposition 1. *[15] The hyperplanes $H_1, H_2, ..., H_N$ separate the sets S_1 and S_2 in the sense of N-polyhedral separability through some Boolean formula if and only if for every pair of points, $p \in S_1$ and $q \in S_2$ there exists an l $(1 \leq l \leq N)$ such that p and q lie on different sides of the hyperplane H_l.*

Proposition 2. *[15] Suppose A and B are sets of points in R^d with integer coordinates, and suppose there exists a hyperplane $H = \{z \in R^d | y^Tz = y_0\}$ that separates A from B. Assuming $y^Ta < y_0$ for $a \in A$. Under these conditions, there exist a hyperplane $H = \{z \in R^d | z^Tx = x_0\}$, a positive rational number r, and integers j_A, j_B $(j_A, j_B \geq 1, \quad j_A + j_B \leq d+1)$ such that:*

(i) *For every $a \in A$, $x^Ta \leq x_0 - r$.*

(ii) *For every $b \in B$, $x^Tb \geq x_0 + r$.*

(iii) *For at least j_A points $a \in A$, $\quad x^Ta = x_0 - r$, and for at least j_B points $b \in B, x^Tb = x_0 + r$.*

We briefly describe the algorithm of Megiddo [15].

If two sets S_1 and S_2 are separable with N hyperplanes then there exists N pairs of complementary subsets $A_i, B_i \subseteq S_1 \cup S_2$ and N hyperplanes H_i such that H_i separates A_i from B_i. From proposition 2, there exist such hyperplanes that

satisfy equalities stated there and, moreover, the separating hyperplanes can be chosen from a finite set. Each of the hyperplanes is determined by a set of at most $d+1$ points [15]. All such hyperplanes are computed; there are at most $O(n^{d+1})$ such planes, and computation of each plane can be done in polynomial time. Each N combination of such planes is checked to see if it separates S_1 and S_2; there are $O(n^{N(d+1)})$ such combinations and the separability property of each combination can be checked in polynomial time.

Let $O(n^p)$ be the time complexity of the linear programming problem. The cost of computing the required hyperplanes is given by $O(n^{d+p+1})$. Given N hyperplanes, its N-polyhedral separability can be checked in $O(n^2N)$ time using the proposition 1. Thus the complexity of the above algorithm is $O(n^{d+p+1} + Nn^{N(d+1)+2}) = O(Nn^{N(d+1)+2})$, by treating d as a constant. We now present an algorithm that has a smaller exponent for n. Our algorithm works as follows:

(a) Compute the Voronoi diagram of the n points of $S_1 \cup S_2$;

(b) Compute the d-faces of the Voronoi diagram that are shared by points from both S_1 and S_2. These faces partition $\Re^d$ such that each partition contains points from only S_1 or only from S_2.

(c) Compute the convex hull of points of each of the partition of (b).

  (i) For each pair of non-intersecting hulls compute a separating hyperplanes as described in Section 3.1.

  (ii) For each pair of intersecting hulls compute a set of hyperplanes corresponding to each 0-face in step (a). Note that each set of this 0-face is generated by $d+1$ points from $S_1 \cup S_2$ with at least one point from each of S_1 and S_2.

(d) For every set of N hyperplanes chosen from step (c), check if this set correctly separates S_1 and S_2.

The steps (a) and (b) are computed using the algorithm of Edelsbrunner [9] that runs in $O(n \log n)$ time if $d \leq 2$ and in $O(n^{\lceil (d+1)/2 \rceil})$ time for $d \geq 2$; For step (c), we note that there are $O(n^{\lceil d/2 \rceil})$ 0-faces in the Voronoi diagram. It is direct to show that the complexity of this step is $O(n^{\lceil (d+1)/2 \rceil})$. By noting that there will be a set of size $O(n^{\lceil (d+1)/2 \rceil})$ from which N hyperplanes are chosen in step (d), we have the complexity of this step given by $O(n^{N\lceil (d+1)/2 \rceil + 2})$.

Thus we have the following theorem.

Theorem 3. *The separating function for two N-polyhedral separable half-spaces can be computed in $O(n^{\lceil (d+1)/2 \rceil})$ time.*

4. ON-LINE VERSION

Now we address the problem of training the N perceptrons as the examples come in, i.e. on-line, much in the spirit of perceptron learning procedure of Nilsson [18]. In this section we present an analogous learning procedure for the case of N-perceptrons.

4.1 MULTI-PERCEPTRON THEOREM

Let the training sequence be denoted by $S_X : x_1, x_2, ...$ such that x_i is specified to be either in S_1 or in S_2. Each $x \in S_i$ is infinitely repeated in S_X (as assumed in the case of classical perceptrons formulation [18]). We initiate N-perceptrons with arbitrary weights. Let W_i^k denote the weight of the ith perceptron after the kth step, let $H_i^k : \Re^d \mapsto \{0,1\}$ denote classification function of the ith perceptron at kth step. i.e., $H_i^k(x) = sign\left((W_i^k)^T x\right)$ for $x \in \Re^d$, where $sign : \Re \mapsto \{0,1\}$ such that $sign(a) = 1$ if $a > 0$ and $sign(a) = 0$ if $a < 0$. Recall that N hyperplanes (the classification functions of perceptrons) partition $\Re^d$ into cells such that in each cell H_i^k is invariant at all points of the cell.

The outline of the training procedure is as follows: Given $x_k \in S_1$ (the case $x_k \in S_2$ is similar), we obtain the Boolean vector

$$\beta^k(x) = (H_1^k(x), H_2^k(x), \ldots, H_N^k(x))$$

of the outputs of the N-perceptrons.

Recall that each Boolean vector of the form $\beta^k(x)$ defines a cell in $\Re^d$. Then we have the following cases:

(a) For all $y \in S_2$ that have been encountered in the sample, $\beta^k(x_k) \neq \beta^k(y)$; for each $y \in S_2$, there exists at least one hyperplane that yields a different component of $\beta^k()$ for y and x_k;

(b) There exists $y \in S_2$ encountered in the training sample such that $\beta^k(y) = \beta^k(x_k)$, i.e. every perceptron is unable to distinguish between y and x_k.

We then have the following learning algorithm:

For case (a) $W_i^{k+1} = W_i^k$ for $i = 1, 2, \ldots, N$.

For case (b), we have two subcases for W_i^k, one for each H_i^k that corresponds to the cell containing both x_i and y:

$$W_i^{k+1} = W_i^k + cx_i, \quad \text{if} \quad (W_i^k)^T x_k < 0$$

$$W_i^{k+1} = W_i^k - cx_i, \quad \text{if} \quad (W_i^k)^T x_k > 0$$

where c is a constant.

We now establish that the above learning procedure converges in that there exist some $k_0 < \infty$ such that $W_i^{k_0} = W_i^{k_0+1} = W_i^{k_0+2} = \ldots$, and for $x \in S_1$ and $y \in S_2$, we have $\beta^k(x) \neq \beta^k(y)$ for $k \geq k_0$.

Theorem 4. *The separating function for two finite N-polyhedral separable sets S_1 and S_2 computed by the incremental algorithm converges to the actual one in a finite number of steps; each step has a complexity polynomial in $|S_1 \cup S_2|$.*

Proof: We show that the learning problem for N-polyhedral separability is equivalent to that of linear containment in a higher dimensional space. We say that a set $S \subseteq \Re^d$ is *linearly contained* if there exists a hyperplane H such that for every $x \in S$, we have $H^T x > 0$. It is direct to see that if S_1 and S_2 are linearly separable, then we can augment them with a last component such that there exists a hyperplane H such that $H^T x_1 > 0$ for all $x_1 \in S_1$, and $H^T x_2 < 0$ for all $x_2 \in S_2$. Let $-S = \{-x | x \in S\}$. If S_1 and S_2 are linearly separable then $S_1 \cup -S_2$ is linearly contained.

Now consider that S_1 and S_2 are N-polyhedral separable. S_1 (S_2) can be partitioned into N blocks $A_1, A_2, \ldots, A_N$ ($B_1, B_2, \ldots, B_N$) such that every pair A_i and B_j is linearly separable. Moreover the elements of S_1 and S_2 can be distributed into cells formed by the hyperplanes $H_1, H_2, \ldots, H_N$ such that each cell contains only elements of S_1, or only elements of S_2, or no elements from $S_1 \cup S_2$. We now generate a set $Z \subset \Re^{dNf_d^d(N)}$,

$$f_d^d(N) = f_k^d(N) = \sum_{i=0}^{k} \binom{d-i}{k-i} \binom{N}{d-i}$$

such that Z is linearly contained in $\Re^{dNf_d^d(N)}$.

For each $x \in S_1$, we generate vectors of the form $z \in Z$ as follows: Recall that the N hyperplanes partition $\Re^d$ into $f_d^d(N)$ cells. Imagine that $z \in Z$ as juxtaposition of vectors of size Nd for each cell; each cell C is represented by a vector C_v of size Nd such that the first d components correspond to H_1 and form 1st block; next d components correspond to H_2, and form 2nd block; and, so on. Consider the cell C in which x_k is contained. The N blocks of cell C correspond to each of H_i. The vector $z_{x_k}^i$ corresponding to H_i is formed by storing x_k or $-x_k$ in the ith block of C_v. Corresponding to H_j, we store x_k if $(H_j)^T x_k > 0$ and store $-x$ if $(H_j)^T x_k < 0$. We store 0's in all other components components of z. We repeat the same process for $x \in S_2$.

Now S_1 and S_2 are N-polyhedral separable if and only if Z obtained as above is polyhedrally contained such that the plane containing z is obtainted by juxtaposing an (Nd)-vector $f_d^d(N)$ times. To see this, consider that S_1 and S_2 are N-polyhedral separable, and $H_1, H_2, \ldots H_N$ be the corresponding separating planes. Then consider the vector $\hat{z} \in \Re^{dNf_d^d(N)}$ formed by concatenating the sequence of separating planes for each cell, i.e. $\hat{z}$ is concatenation of $< H_1, H_2, \ldots, H_N >$ for $f_d^d(N)$ times. It is clear that this plane will contain all the vectors of Z. On the other hand, the required plane containing Z can be decomposed into suitable parts to obtain the separating surfaces for S_1 and S_2.

The proof is complete by noting that the classical perceptron learning algorithm of Z exactly corresponds to the above proposed learning algorithm for N-perceptrons. QED.

5. CONCLUSIONS

Given two subsets S_1 and S_2 (not necessarily finite) of $\Re^n$ separable by a Boolean combination of N halfspaces, we consider the problem of learning (in the sense of Valiant) the separation function from a set of finite number of examples, i.e. we produce with high probability a function close to the actual separating function. Our solution consists of a system of N perceptrons and a single consolidator which combines the outputs of the individual perceptrons. We show that an off-line version of this problem, where the examples are given in a batch, can be solved in polynomial time. We also provide an on-line learning algorithm that incrementally solves the problem by suitably training a system of perceptrons in the spirit of classical perceptron learning algorithm. Our solutions to the above problems constitute examples of a system of N-learners (discussed in [21]) capable of accomplishing a task that is not achievable by a single learner, for a single perceptron is incapable of separating sets that are not linearly separable.

REFERENCES

1. J. Amsterdam, Extending the Valiant learning model, *Proc. 5th Int. Conf. on Machine Learning*, 1988, 381-394.
2. D. Angluin and P.D. Laird, Learning from noisy examples, *Machine Learning*, vol. 2, 1988, 343-370.
3. E.B. Baum, On learning a union of half spaces, *Journal of Complexity*, vol.6, 1990, pp. 67-101.
4. G. Benedek and A. Itai, Nonuniform learnability, *Proc. of 15th Int. Conf. on Automata, Languages and Programming*, 1988, 82-92.
5. J.L. Bentley and M.I. Shamos, Divide-and-conquer in multidimensional space, *Proc. 8th ACM Annual Symp. on Theory of Computing*, 1976, pp. 220-230.
6. A. Blumer, A. Ehrenfeucht, D. Haussler and M. Warmuth, Learnability and the Vapnik-Chervonenkis Dimension, *J. Assoc. Comput. Mach.*, vol 36(4), 1989.
7. T. Cover, Geometrical and statistical properties of systems of linear inequalities with applications to pattern recognition, *IEEE Trans. Elect. Comput.*, vol.14, 1965, pp. 326-334.
8. R.M. Dudley, A course on empirical processes, in *Lecture Notes in Mathematics*, vol. 1097, Springer-Verlag, New York, 1984.
9. H. Edelsbrunner, *Algorithms in Combinatorial Geometry*, Springer-Verlag, 1987.
10. D. Hausler, Generalizing the PAC model: sample size bounds from metric dimension-based uniform convergence, *Proc. 3rd Symp. on Foundations of Computer Science*, 1989, 40-45.
11. M. Kallay, Convex hull algorithms in higher dimensions, unpublished manuscript, Dept. Mathematics, Univ. Oklahoma, Norman, Oklahoma, 1981.
12. N. Karmarkar, A new polynomial-time algorithm for linear programming, *Combinatorica*, vol.4, 1984, 373-395.
13. L.G. Khachiyan, Polynomial algorithm for linear programming, *Dokl. Akad. Nauk SSSR*, vol. 244, 1979, pp. 1093-1096 (in Russian).
14. N. Littlestone, Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm, *Machine Learning*, vol. 2, 1988, pp. 285-318.
15. N. Megiddo, On the complexity of polyhedral separability, *Discrete and Computational Geometry*, vol.3, 1988, pp. 325-337.
16. M. L. Minsky and S. A. Papert, *Perceptrons*, MIT Press, Cambridge, Mass., Second printing 1988.
17. B.K. Natarajan, On learning sets and functions, *Machine Learning*, vol 4, 1989, 67-97.
18. N. J. Nilsson, *Learning Machines*, McGraw-Hill Book Co., 1965.

19. C.H. Papadimitriou, and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall Inc., New Jersey, 1982.

20. F.P. Preparata, and M.I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, 1985.

21. N.S.V. Rao, E.M. Oblow, C.W. Glover and G.E. Liepins, N-learners problem: Fusion of Concepts, Tech. Rep. ORNL/TM-11904, (CESAR-91/23),Oak Ridge National Laboratory, 1991.

22. J. Slansky and G. N. Wassel, *Pattern Classifiers and Trainable Machines*, Springer-Verlag, New York, 1981.

23. L.G. Valiant, A theory of the learnable, *Communications of the ACM*, 27(11):1134-1142.

24. V.N. Vapnik, *Estimation of Dependences Based on Empirical Data*, Springer-Verlag, New York, 1982.

25. Y. Ye, An extension of Karmarkar's algorithm and the trust region method for quadratic programming, in *Progress in Mathematical Programming: Interior-Point and Related Methods*, Ed. N. Megiddo, Springer-Verlag, New York, 1989, pp. 49-63.

INTERNAL DISTRIBUTION

1. B. R. Appleton
2. M. Beckerman
3–7. C. W. Glover
8. J. P. Jones
9–13. R. C. Ward
14–18. R. C. Mann
19–23. E. M. Oblow
24. F. G. Pin
25. V. Protopopescu
26. D. B. Reister
27. P. Spelt
28–32. F. J. Sweeney
33. R. W. Brockett (Consultant)
34. J. J. Dorning (Consultant)
35. J. E. Leiss (Consultant)
36. N. Moray (Consultant)
37. M. F. Wheeler (Consultant)
38. EPMD Reports Office
39–40. Laboratory Records Department
41. Laboratory Records, ORNL–RC
42. Document Reference Section
43. Central Research Library
44. ORNL Patent Section

EXTERNAL DISTRIBUTION

45. Dana Angluin, Computer Science Department, Yale University, P. O. Box 2158, New Haven, Conecticut 06520
46. Peter Allen, Department of Computer Science, 450 Computer Science, Columbia University, New York, N.Y.
47. Jacob Barhen, Jet Propulsion Laboratory, MS 300/310, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, California 91109-8099
48. Eric Baum, NEC Research Institute, 4 Independence Way, Princeton, NJ 08540
49. Anselm Blumer, Department of Mathematics & Computer Science, Tufts University, Medford, Massachusetts 02155
50. Wayne Book, Department of Mechanical Engineering, J S Coon Bldg, Room 306, Georgia Institute of Technology, Atlanta, GA 30332
51. Peter Cheeseman, NASA Ames Research Center, Mail Stop 244-17, Moffett Field, California 94035
52. Jie Cheng, Department of Electrical Engineering & Computer Science, The University of Michigan, Ann Arbor, Michigan 48109-2122
53. Steve Dubowsky, MIT, Bldg 3, Rm 469A, 77 Massachusetts Ave, Cambridge, MA 02139
54. Andrzej Ehrenfeucht, University of Colorado, Boulder, Colorado 80309
55. Jerome Friedman, Deartment of Statistics, Stanford University, Sequaia Hall, Stanford, CA 94305
56. Irwin. R. Goodman, Naval Ocean Systems Center, San Diego, CA 92152
57. John J. Grefenstette, Navy Center for Applied Research in Artificial Intelligence, Naval Research Laboratory, Washington, D. C. 20375-5000
58. David Haussler, Department of Computer & Information Sciences, University of California, Santa Cruz, California 95064

59. Haym Hirsh, Computer Science Department, Stanford University, Stanford, California 94305
60. Charles C. Jorgensen, Computational Systems Research, Ames Research Center, Mailstop 269-3, Code FIC, Moffett Field, CA 94035
61. Avi Kak, Department of Electrical Engineering, Engineering Mall, Purdue University, Lafeyette, IN 47907
62. Michael Kearns, Aiken Computation Laboratory, Harvard University, Cambridge, Massachusetts 02138
63. M. H. Kalos, Courant Institute of Math Science, New York University, 251 Mercer Street, New York, NY 10012
64. R. C. Lacher, Department of Computer Science, 206 Love B-173, Tallahassee, FL 32306-4019
65. Philip D. Laird, NASA Ames Research Center, Moffett Field, California 94035
66. Pat Langley, Department of Information & Computer Science, University of California, Irvine, California 92717
67. Nick Littlestone, Department of Computer & Information Sciences, University of California, Santa Cruz, California 95064
68. Oscar P. Manley, U.S. Department of Energy, Division of Engineering, Mathematical and Geosciences, Office of Basic Energy Sciences, Germantown, MD 20545
69. Alan Meyrowitz, Office of Naval Research, Code 1133, 800 N. Quincy St., Arlington, VA 22217
70. B. K. Natarajan, The Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213
71. Judea Pearl, Computer Science Department, University of California, 405 Hilgard Ave., Los Angeles, CA 90024
72. J. R. Quinlan, Basser Department of Computer Science, University of Sydney, Sydney, New South Wales 2006, AUSTRALIA

73–77. N. S. V. Rao, Department of Computer Science, Old Dominion University, Norfolk, VA 23529-0162

78. Zbigniew W. Ras, Department of Computer Science, University of North Carolina, Charlotte, North Carolina 28223
79. Ronald L. Rivest, MIT Laboratory for Computer Science, Cambridge, Massachusetts 02139
80. George Shackelford, Department of Information & Computer Science, University of California, Irvine, California 92717
81. Robert Sloan, MIT Laboratory for Computer Science, Cambridge, Massachusetts 02139
82. Wes Snyder, Department of Radiology, Bowman Gray School of Medicine, 300 South Hawthorne Drive, Winston Salem, NC 27103
83. Richard S. Sutton, GTE Laboratories, Inc., 40 Sylvan Road, Waltham, Massachusetts 02254
84. V. R. R. Uppuluri, 130 Indian Lane, Oak Ridge, TN 37830
85. Paul E. Utgoff, Department of Computer & Information Science, University of Massachusetts, Amherst, Massachusetts 01003
86. L. G. Valiant, Aiken Computation Laboratory, Harvard University, Cambridge, Massachusetts 02138
87. V. N. Vapnik, AT&T Bell Labs, Room 4G314, Crawfords Corner Rd., Holmdel, NJ 07733

88. Jeffrey S. Vitter, Department of Computer Science, Brown University, Providence, Rhode Island 02912
89. Manfred K. Warmuth, Department of Computer & Information Sciences, University of California, Santa Cruz, California 95064
90. C. R. Weisbin, Jet Propulsion Laboratory, MS 198/330, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, California 91109-8099
91. Lotfi A. Zadeh, Computer Science Division, University of California, Berkeley, CA 94720
92. Maria Zemankova, Department of Computer Science, University of Tennessee, Knoxville, Tennessee 37996-1301
93. Office of Assistant Manager, Energy Research and Development, DOE-ORO, P.O. Box 2001, Oak Ridge, TN 37831

94–103. Office of Scientific and Technical Information, Department of Energy, Oak Ridge, TN 37831