REPRESENTING KNOWLEDGE IN LEARNING SYSTEMS BY PSEUDO BOOLEAN FUNCTIONS

Haim Shvaytser *

Department of Computer Science
Columbia University
New York, NY 10027

Abstract

Concepts that can be expressed as solutions to multilinear pseudo boolean equations with a bounded degree are shown to be learnable in polynomial time from positive examples. This implies the learnability from positive examples of many families of boolean formulae by a unified algorithm. Some of these formulae were not previously known to be learnable, and some were known to be learnable by different algorithms.

* Author's current address: Department of Computer Science, Cornell University, Ithaca, NY 14853.
The author has been partially supported by the Chaim Weizmann Postdoctoral fellowship.

1. Introduction

A complexity based theory of the learnable was introduced by Valiant in [1]. One of the motivations of the suggested theory was "to shed light on the boundary between the classes of expressions that are learnable in polynomial time and those that are not, suggesting concrete principles for designing realistic learning systems." The results obtained in [1] and related works [2-4] are mainly about the learnability of boolean formulae, using the propositional calculus for representing the learned knowledge. These results characterize several classes of boolean functions as learnable, and several others as not learnable.

Many of the negative results, about classes of boolean formulae that cannot be learned, were obtained with respect to specific representations of the knowledge in the learning algorithm. Examples are the results of Natarajan, in [3], which show that the class of boolean functions that can be learned from positive examples using the same class of functions for representing the learned knowledge is severely limited, and the results in [4], that characterize many families of boolean formulae as non learnable unless $RP = NP$. As a specific example, consider boolean formulae in disjunctive normal form (DNF), with a small (bounded) number of terms. It was shown in [4] that unless $RP = NP$, such formulae cannot be learned if the knowledge representation is also chosen to be DNF with the same (or even twice as many) number of terms. However, such formulae can be easily learned if the knowledge representation is chosen to be boolean formulae in conjunctive normal form with a bounded size of conjuncts ($k-CNF$) [5].

The work presented in this paper exemplifies that some of the negative results about learnability are to be attributed to the specific knowledge representation scheme, not necessarily implying inherent difficulty in learning the concepts. We investigate the representation of knowledge in terms of pseudo boolean equations. We show that concepts that can be expressed as solutions to multilinear equations with bounded degree are learnable with one sided error from positive examples only. The family of concepts that can be represented and learned in this way includes $k-CNF$, $k-term-DNF$ (that was shown not to be learnable by $k-term-DNF$ in [4]), and a restricted version of $k-DNF$, that by using

the results of [3] can be identified as not learnable from positive examples, using k–DNF as the scheme of knowledge representation.

2. Definitions

We consider n boolean variables $x_1, \cdots, x_n$, each of which can take the value 0 or 1. A *vector* v is an assignment to each of the n variables of a value from $\{0,1\}$. A *concept* is a subset of the 2^n vectors, where each vector in this subset is a *positive example*, and all the rest are *negative examples* of the concept.

The definition of learnability of a concept by a class of representations appears in [4]. We use the same definition, but consider learnability from positive examples only. The learning process can be viewed as follows: a concept from a known class of concepts is chosen by an oracle that also fixes a probability distribution over its positive examples. The learner draws (positive) examples according to the probability distribution. A learning algorithm is applied to these examples, and produces as output an approximate representation of the concept. This representation is tested on both positive and negative examples. The positive examples for the test are chosen from the same probability distribution as in the learning phase, but the distribution of the negative examples in the test is unknown (determined by the oracle). The learning algorithm is required to guarantee with arbitrary high probability arbitrary high score in the test. We observe that this implies that no mistake can be permitted in classifying negative examples as being positive.

The following is a formal definition: *A class C of concepts, each of which is of size polynomial in n, is learnable from positive examples by a class of representations G if there exists an algorithm that for any $h > 1$, for any $c \in C$, and for any probability distribution D over the positive examples of c*

(a) The algorithm gets as input a number of positive examples polynomial in both the adjustable parameter h, and n. The examples are obtained by sampling from the probability distribution D.

(b) The algorithm runs in time polynomial in both h and n.

(c) The output of the algorithm is a representation $g \in G$ such that:

(i) $g(v) = TRUE \implies v$ *is a positive example of* c.

(ii) with probability of at least $(1 - \frac{1}{h})$ $\sum_{\substack{v \text{ positive example} \\ g(v) = FALSE}} D(v) < \frac{1}{h}$.

A *pseudo boolean function* f is any mapping from the set of 2^n vectors to the real axis. A concept is *represented by a set of d pseudo boolean functions* $f_1, \cdots, f_d$ if $f_i(v) = 0 \quad i=1, \cdots, d$ whenever v is a positive example, and $\exists i \quad f_i(v) \neq 0$ whenever v is a negative example. In other

words, the concept is the set of vectors that are solutions to the system of equations $f_i(x_1, \cdots, x_n) = 0 \quad i=1, \cdots, d$.

A pseudo boolean function f is *consistent with a concept* if $f(v) = 0$ whenever v is a positive example. A class C of boolean formulae over n variables is represented by multilinear equations of degree k if every $c \in C$ can be represented by a k degree multilinear equation. (In general, k is a function of n.) For example, the boolean formula $x_1 \vee (x_2 \wedge x_3)$ can be represented by the second degree (bilinear) multilinear equation: $x_1x_2 + x_1x_3 - 2x_1 - x_2 - x_3 + 2 = 0$.

The *information* of a pseudo boolean function (with respect to a concept) is defined as the set of negative examples which it can detect, i.e.,

$$\text{Information}(f) = \{v \ ; f(v) \neq 0\}.$$

We observe that a concept is represented by the pseudo boolean functions $\{f_I\}$ if and only if the functions $\{f_I\}$ are all consistent with the concept, and their combined information, $\bigcup_I \text{Information}(f_I)$, is the set of all negative examples. Finally, we say that the function f_0 can be derived from the set of consistent functions $f_1, \cdots, f_k$ if $\text{Information}(f_0) \subset \bigcup_{i=1}^{k} \text{Information}(f_i)$.

3. Algebraic structure of concepts expressed by pseudo boolean functions

A multilinear function in the variables $x_1, \cdots, x_n$ with a degree k, is a function of the type: $M(x_1, \cdots, x_n) = \sum_{i=0}^{k} \sum_{j_1 < \cdots < j_i} a_{j_0 \cdots j_i} \cdot x_{j_1} \cdot \ldots \cdot x_{j_i}$. Each multilinear function can be viewed as a linear combination of its monomials $x_{j_1} \cdot \ldots \cdot x_{j_i}$. We will use the variables $\{y_i\}$ to denote these monomials. For example, $2 \cdot x_1x_2 + 2 \cdot x_1x_3 + 4 \cdot x_4 - 5$ is a bilinear function (k=2), expressed as a linear combination of the monomials $\{x_1x_2 \ , \ x_1x_3 \ , \ x_4 \ , \ 1\}$. The set of all monomials $\{y_i\}$ up to a degree k has $\sum_{i=0}^{k} \binom{n}{i}$ elements. The representation of a multilinear function as a linear combination of monomials is called *the canonical representation.* Notice that any pseudo boolean function can be expressed as a multilinear function.

We say that two functions are *identical modulo a concept* if they have the same values for all positive examples of the concept.

Theorem 1: The set of functions modulo a concept with r positive examples is a vector space (over $\mathbb{R}$) with a basis of r multilinear functions.

Proof: The set of functions is a vector space under the standard definition of addition and multiplication by scalars from $\mathbb{R}$. To determine its dimension modulo the concept, let the set of positive examples be $\{v_1, \cdots, v_r\}$. For each positive example $v_i = (x_1^i, \cdots, x_n^i)$ we define the following multilinear function:

$$P_i = \prod_{x_\alpha^i=1} x_\alpha \cdot \prod_{x_\beta^i=0} (1-x_\beta) \tag{1}$$

For example, if $v_i = (1,0,0,1)$, $P_i = x_1 \cdot (1-x_2) \cdot (1-x_3) \cdot x_4$.

Clearly, $P_i(v_j) = \begin{cases} 1 & i=j \\ 0 & i \neq j \end{cases}$ and we maintain that $\{P_1, \cdots, P_r\}$ is a basis for the vector space of all functions modulo the given concept. We first show that it spans the set of all functions. Let $f(x_1, \cdots, x_n)$ be an arbitrary function, and

$$f(v_i) = f(x_1^i, \cdots, x_n^i) = \alpha_i \qquad i=1, \cdots, r \ .$$

The following linear combination of $\{P_i\}$ is identical to f modulo the concept:

$$M(x_1, \cdots, x_n) = \sum_{i=1}^{r} \alpha_i \cdot P_i \ .$$

To see that $\{P_i\}$ are linearly independent modulo the concept, consider an arbitrary linear combination that identically equals zero modulo the concept (i.e., equals zero for all *positive* examples):

$$a_1 P_1 + \cdots + a_r P_r = 0$$

Substituting the values of $P_1 \cdots P_r$ for the example v_i reduces the above equation to $a_i = 0$ □.

Corollary 1.1: There is a basis of r monomials $\{y_i\}$ to the set of all functions modulo a concept with r positive examples.

Corollary 1.2: The canonical representation of a multilinear function is uniquely determined by the function values for all 2^n examples (all the positive and negative examples).

Proof: By applying the above corollary to the concept for which all the examples are positive examples ($r=2^n$), the set of all 2^n monomials $\{y_i\}$ is a basis, and the corollary follows from the uniqueness of the representation in a basis. □

3.1. A basis of pseudo boolean functions

The next two theorems will characterize the set of consistent functions as a finite dimensional vector space. The advantage of this representation is that it is relatively easy to determine a basis to the vector space of consistent functions. Although a basis to this vector space is not necessarily a minimal set of consistent functions, we will show that the number of functions in a basis is polynomial when one is concerned only with concepts that can be represented by bounded degree multilinear equations.

Theorem 2: Let F be a set of arbitrary functions of n binary variables. The set of all consistent functions modulo a concept with r positive examples that can be expressed as *linear* combinations of functions from F is a vector space of a finite dimension $d=v-w$, and $0 \leq d \leq 2^n - r$, where $v = dim\ Span(F)$ and $w = dim\ Span(F)$ *modulo the concept.*

Proof: Let $V = Span(F)$ and $W = Span(F)$ *modulo the concept* . V can be viewed as $Span(F)$ modulo the concept with 2^n positive examples. The difference between V and W is that two different functions in V may be identical modulo the concept, and therefore, the same function in W (that is, when they have the same values for the r positive examples). Clearly, $W \subset V$. Let $P : V \rightarrow W$ be the projection from V into W. P is linear, onto, and its kernel is all linear combinations of functions that are mapped into "0" in W, i.e., the required consistent functions. Therefore, the set of consistent functions is a vector space, and because dim Im $P + dim\ Ker\ P = dim\ V$, we have $w + d = v$. Since $W \subset V$, $d = v-w \geq 0$. To see that $d \leq 2^n - r$, let $\tilde{F}$ be a set of 2^n-v independent functions, also independent of the functions in F. Let $\tilde{v} = dim\ Span(F \bigcup \tilde{F})$ and $\tilde{w} = dim\ Span(F \bigcup \tilde{F})$ *modulo the concept* . We have: $\tilde{v} = v + 2^n-v = 2^n$, and $\tilde{w} \leq w + 2^n - v$. But from Theorem 1, $\tilde{w} = r$, and therefore $v - w \leq 2^n-r$ □

Theorem 3: For a concept with r positive examples, let $F = \{f_1, \cdots, f_t\}$ be an arbitrary set of t functions. Denote by $B \subset F$ a basis to the vector space of the functions in F modulo the concept. We will assume without loss of generality that $B = \{f_1, \cdots, f_w\}$, where $w = dim\ Span(F)$ *modulo the concept.* Let $f_i = \sum_{j=1}^{w} a_{ij} \cdot f_j \quad i=w+1, \cdots, t$ be the linear dependency relations modulo the concept, then *all* the consistent functions that can be expressed as linear combinations of $\{f_1, \cdots, f_t\}$ are derivable from the $t-w$ consistent functions:

$$f_i - \sum_{j=1}^{w} a_{ij} \cdot f_j \qquad i=w+1, \cdots, t \tag{2}$$

Proof: It is enough to consider the case in which $\{f_1, \cdots, f_t\}$ are linearly independent. In this case, $t = dim\ Span(F)$, and from Theorem 2 the set of all consistent functions that can be expressed as linear combinations of $\{f_1, \cdots, f_t\}$ is derivable from a basis of $t-w$ consistent functions. Therefore, it is enough to prove that the $t-w$ consistent functions (2) are linearly independent. Consider an arbitrary linear combination of the functions (2) that equals zero:

$$\sum_{i=w+1}^{t} b_i (f_i - \sum_{j=1}^{w} a_{ij} \cdot f_j) = 0 \tag{3}$$

To prove the theorem we have to show that $b_i = 0 \quad i=w+1, \cdots, t$. The left side of (3) is a linear combination of functions $f_i \quad i=1, \cdots, t$, and can be expressed as $\sum_{i=1}^{t} \alpha_i \cdot f_i = 0$, with $\alpha_i = b_i \quad i=w+1, \cdots, t$. Since $\{f_i\}$ are linearly independent, $\alpha_i = 0 \quad i=1, \cdots, t$, and therefore, $b_i = 0 \quad i=w+1, \cdots, t$. □

Corollary 3.1: All consistent functions modulo a concept that can be expressed as multilinear functions with a bounded degree are derivable from a polynomial set of consistent functions.

Proof: For all monomials with degree not exceeding k, the value of t in Theorem 3 is $\sum_{i=0}^{k} \binom{n}{i}$.

□

4. An algorithm for generating a basis of consistent functions

In this section we describe an algorithm that generates the consistent functions (2). Given a set of functions $\{f_i\}$, and a set of positive examples, the algorithm generates a basis for the set of consistent functions that can be expressed as linear combinations of functions from $\{f_i\}$. For bounded degree multilinear functions the algorithm is polynomial. Furthermore, in this case the consistent functions (2) are multilinear functions with bounded degree in their canonical representation. Therefore, each one of them can be computed in polynomial time.

As will be shown, the information we require to determine the consistent functions is the following statistics:

$$R_{pq} = \sum_{v_i} f_p^i \cdot f_q^i \cdot D(v_i) \ . \tag{4}$$

To get a basis for the set of consistent functions that can be expressed as linear combinations of functions in the span of $F = \{f_1, \cdots, f_t\}$ we require the following matrix:

$$R = \begin{bmatrix} R_{1,1} \cdots R_{1,t} \\ \cdots \quad \cdots \quad \cdots \\ R_{t,1} \cdots R_{t,t} \end{bmatrix}$$

This matrix is referred to as *the co–occurrence matrix*. It has the following properties:

a) R is symmetric since by its definition $R_{ij} \equiv R_{ji}$.

b) If $f_1, \cdots, f_t$ are independent modulo the concept, R is positive definite (and therefore, non-singular).

Proof: See [6].

c) If $f_1, \cdots, f_t$ are linearly dependent modulo the concept, then R is singular.

Proof: Since $f_1, \cdots, f_t$ are dependent, there exist coefficients $a_1, \cdots, a_t$ not identically zero such that

$$a_1 \cdot f_1^i + \cdots + a_t \cdot f_t^i = 0 \qquad \forall \textit{positive examples } v_i \tag{5}$$

Multiplying the above equation by $f_j^i \cdot D(v_i)$ and summing over all examples we get:

$$a_1 \cdot R_{1j} + \cdots + a_t \cdot R_{tj} = 0 \qquad j=1, \cdots, t \qquad \square$$

d) If $f_1, \cdots, f_{t-1}$ are independent modulo the concept, but $f_1, \cdots, f_t$ are dependent modulo the concept, then

$$f_t^i - (a_1 \cdot f_1^i + \cdots + a_{t-1} \cdot f_{t-1}^i) = 0 \quad \forall \textit{positive examples } v_i \tag{6}$$

and $a_1 \cdots a_{t-1}$ can be obtained from the system of equations

$$R \cdot \begin{bmatrix} a_1 \\ \cdot \\ \cdot \\ \cdot \\ a_{t-1} \end{bmatrix} = \begin{bmatrix} R_{1,t} \\ \cdot \\ \cdot \\ \cdot \\ R_{t-1,t} \end{bmatrix} \tag{7}$$

(In (7), R is the co-occurrence matrix of $f_1, \cdots, f_{t-1}$.)

Proof: Because $f_1 \cdots f_{t-1}$ are independent, coefficients a_i always exist in (5) such that $a_t = -1$. □

Using the properties of the co-occurrence matrix we describe an algorithm to generate a basis of consistent functions. The algorithm runs identical phases, in which it considers a set of functions, and generates a basis to the set of all consistent functions that can be expressed by linear combinations of them. The information used by the algorithm is the values of R_{pq} given by Equation (4). In the algorithm, I is a set of linearly independent functions modulo the concept, and R is their co-occurrence matrix.

Algorithm A:

Initially, $I = \{f_1\}$, and R is (R_{11}), a matrix of size 1×1.

For each new function f_t:

1- Get the statistics $R_{jt} = \sum_{v_i} f_j^i \cdot f_t^i \cdot D(v_i)$ for all functions $f_j \in I$, and construct the co-occurrence matrix R of the functions in $I \cup \{f_t\}$.

2- If R is singular, solve the system of equations (7), and output the consistent function (6); otherwise, $I \leftarrow I \cup \{f_t\}$.

When the functions in the algorithm are chosen as the set of monomials with bounded degree, the number of phases is polynomial in n (because there is only a polynomial number of monomials), and since the solution of a system of linear equations is also polynomial, the whole algorithm is polynomial. Yet, a more efficient algorithm exists, based on the Choleskey decomposition for positive definite

matrices. This algorithm does not require solving the whole system of equations in each iteration. Instead, it builds a lower triangular matrix Z such that $Z \cdot Z^T = R$.

Algorithm A':

Initially, $I = \{f_1\}$, and Z is $\sqrt{R_{11}}$. u and u_j are auxiliary variables.

For each new function f_t:

get the statistics $R_{j,t}$ for all $f_j \in I$.

For $j = 1, \cdots, |I|$ $\quad u_j = \dfrac{R_{j,t} - \sum_{i=1}^{j-1} u_i z_{ji}}{z_{jj}}$

$$u = \sqrt{R_{t,t} - \sum_{j=1}^{|I|} u_j^2}$$

If $u \neq 0$, add the row $u_1, u_2, \cdots, u_{|I|}, u$ to Z, i.e.,

$$z_{|I|+1,j} \leftarrow u_j \quad j=1, \cdots, |I| \qquad z_{|I|+1,|I|+1} \leftarrow u$$

and add f_t to I.

If $u = 0$, output the consistent function

$$a_1 f_1 + \cdots + a_{|I|} f_{|I|} - f_t = 0$$

where $a_1 \cdots a_{|I|}$ are determined by forward and backward substitutions from

$$Z\ Z^T \begin{bmatrix} a_1 \\ \cdot \\ \cdot \\ \cdot \\ a_{|I|} \end{bmatrix} = \begin{bmatrix} R_{1,t} \\ \cdot \\ \cdot \\ \cdot \\ R_{t-1,t} \end{bmatrix}$$

We observe that for N functions, the complexity of Algorithm A' is $O(N^3)$.

5. Learnability of multilinear functions

In this section we show that Algorithm A (or A') of the previous section can be used to learn consistent functions from examples in the sense of Valiant. We will construct a polynomial algorithm such that: For any h, and for any concept for which f is a consistent function with a degree bounded by k, and all distributions D over the positive examples, the output of the algorithm is a polynomial

number of multilinear functions $G = \{g\}$ with a degree not exceeding k, such that:

(i) with probability of at least $(1 - \frac{1}{h})$ the functions G approximate consistent functions with an error of at most $\frac{1}{h}$.

(ii) f is derivable from G.

Let $L(h,s)$ be the smallest integer such that in $L(h,s)$ Bernoulli trials, each with probability $\frac{1}{h}$ of success, the probability of having fewer than s successes is less than $\frac{1}{h}$. Valiant shows in [1] that for $s \geq 1$, $h > 1$, $L(h,s) \leq 2 \cdot h \cdot (s + \log_e h)$. As we show, the number of examples needed by our algorithm to learn multilinear consistent functions with degree bounded by k is $N(h,n) = L(h, \sum_{i=0}^{k} \binom{n}{i})$. Clearly, for a bounded k, $N(h,n)$ is polynomial in both h and n. More precisely,

$$N(h,n) = O(h \cdot (\log h + n^k)) .$$

The algorithm that learns all multilinear consistent functions up to degree k is the following:

(1)- Randomly choose $N(h,n)$ examples. They are chosen according to their distribution D.

(2)- Compute the values of *all* monomials $Y = \{y_j\}$ with a degree not exceeding k, for each of the examples chosen in (1).

(3)- Compute the statistics for each pair of monomials y_p, y_q in Y: $R_{pq} = \sum_{i=1}^{N} y_p^i \cdot y_q^i$.

(R_{pq} counts the number of examples for which y_p and y_q both hold.)

(4)- Use Algorithm A (or A'), of the previous section with the statistics obtained in step (3) to generate the approximate consistent functions $G = \{g\}$.

Since N is polynomial, and so is the number of all monomials with a degree not exceeding k, the above algorithm is polynomial in both h and n. (It is, however, exponential in k.)

To see that the algorithm generates approximate consistent functions with the desired probability, let G_j be the set of multilinear functions that would have been obtained from the learning algorithm if steps (2),(3),(4) would have been executed after the j'th example is chosen in step (1). We observe that the number of multilinear consistent functions in G_j is the number of independent monomials that are linearly dependent on other monomials modulo the concept where the j examples chosen in (1) are the only positive examples. Now, if a certain monomial linearly depends on others for the first j examples, it also depends on other monomials for the first j' examples, where $j' \leq j$. Consider a series of N Bernoulli trials in which a success is manifested as discovering that at least one of the monomials linearly dependent on others modulo the concept after the first j examples, is linearly independent modulo the concept after the first $j+1$ examples. We have a success if and only if at least one function in G_j is inconsistent with the $j+1$'th example, i.e.,

$$\exists g \in G_j \quad g(v_{j+1}) \neq 0 .$$

Since our assumption is that f is a multilinear consistent function with a degree k, at least one of the monomials is dependent on others for all examples. Therefore, the number of successes cannot exceed the number of monomials of degree bounded by k. Let X_j denote the probability that the next example (the $j+1$'th) does not agree with the functions in G_j. $X_j = \sum D(v)$ with summation over all examples for which $\exists g \in G_j \; g(v) \neq 0$. Clearly, X_j is monotone decreasing as a function of j. We have to show that

$$Prob(X_N > \frac{1}{h}) \leq \frac{1}{h}$$

and this follows because if $X_N > \frac{1}{h}$, $X_j > \frac{1}{h}$ $j=1 \cdots N$, $N = L(h, \sum_{i=0}^{k} \binom{n}{i})$, and we have a series of N Bernoulli trials, each with probability greater than $\frac{1}{h}$, and with less than $\sum_{i=0}^{k} \binom{n}{i}$ successes.

To see that f is derivable from G, consider a concept where the positive examples are only the examples chosen by the learning algorithm in step (1). Since f is also a consistent function for that concept, f is derivable from G by Theorem 3.

6. Learnability of boolean formulae by pseudo boolean functions

The results of the previous section imply that any family of boolean formulae that can be represented as a set of solutions to bounded degree multilinear equations is learnable from positive examples. The straightforward way of translating boolean expressions into multilinear functions is by using the following relations:

$$\neg x \longleftrightarrow 1 - x \quad ; \quad x \wedge y \longleftrightarrow x \cdot y \quad ; \quad x \vee y \longleftrightarrow x + y - x \cdot y \qquad (8)$$

which transform any boolean formula into a pseudo boolean function that equals 1 for positive examples, and 0 for negative examples.

As a first example for the learning power of the multilinear functions we consider the case of $k-CNF$ formulae, that were shown to be learnable by Valiant in [1]. To learn $k-CNF$ formulae of n variables with error $\frac{1}{h}$, Valiant's algorithm requires $O(h \cdot (\log h + n^{k+1}))$ examples, while our algorithm requires only $O(h \cdot (\log h + n^{k}))$ examples.

Theorem 4: $k-CNF$ is learnable by a multilinear equation of degree bounded by k.

Proof: Let the $k-CNF$ expression be $c_1 \wedge c_2 \wedge \cdots \wedge c_N$. Since each of the clauses contain at most k literals, it follows from (8) that it can be expressed as a multilinear function of degree k. Denote by M_i the multilinear function that corresponds to c_i. $M_i = 1$ if $c_i = TRUE$, and $M_i = 0$ if $c_i = FALSE$. The following multilinear equation of degree k is a representation of the $k-CNF$ expression:

$$\sum_{i=1}^{N} M_i - N = 0 \qquad \square .$$

A $k-term-DNF$ formula is a DNF formula with at most k terms. It is known to be learnable by $k-CNF$ [5]. We give a direct proof for its learnability by multilinear functions.

Theorem 5: $k-term-DNF$ is learnable by a multilinear equation of degree bounded by k.

Proof: Let the $k-term-DNF$ expression be $m_1 \vee m_2 \vee \cdots \vee m_k$, where $m_i = y_1^i \wedge \cdots \wedge y_{t_i}^i$, with $t_i = p_i + n_i$, $y_j^i = x_j^i \quad j = 1 \cdots p_i$, and $y_{j+p_i}^i = \neg x_{j+p_i}^i \quad j = 1 \cdots n_i$. The following is a

representation by a k degree multilinear equation:

$$\prod_{i=1}^{k}(\sum_{j=1}^{p_i} x_j^i - \sum_{j=1}^{n_i} x_{p_i+j}^i - p_i) = 0 \qquad \square$$

Example: $(x_1 \wedge \overline{x}_2 \wedge x_3) \vee (\overline{x}_1 \wedge x_2)$ is represented by $(x_1 + x_3 - x_2 - 2)\cdot(x_2 - x_1 - 1) = 0$.

Consider a subset of N variables out of the n variables $x_1, \cdots, x_n$, and denote its elements by $x_{S_1}, \cdots, x_{S_N}$. A concept is represented by a boolean threshold function if there are numbers N and t such that $\sum_{i=1}^{N} x_{S_i} > t$ $\Longleftrightarrow$ $(x_1, \cdots, x_n)$ *is a positive example.* It was shown in [4] that threshold functions are not learnable. The following theorem shows that a restricted version of the threshold functions is learnable from positive examples.

Theorem 6: With the above terminology, threshold functions for which $N - t \leq k$ are learnable by multilinear equations of degree k.

Proof: The threshold functions under the condition of the theorem are represented by the following multilinear equation: $\prod_{j=1}^{k}(\sum_{i=1}^{N} x_{S_i} - t - j) = 0$ $\square$.

It should be noted that under the conditions of the theorem the threshold function can be expressed as a k+1$-DNF$. Thus, they can be learned by Valiant's algorithm, by using $O(h\cdot(\log h + n^{k+2}))$ examples. Learning them by multilinear equations requires only $O(h\cdot(\log h + n^k))$ examples.

Let $m_1 \vee m_2 \vee \cdots \vee m_N$ be a $k-DNF$ expression, i.e., each of the monomials $\{m_i\}$ has at most k literals. Natarajan showed in [3] that $k-DNF$ expressions are not learnable by $k-DNF$ expressions from positive examples. We consider a restricted version of the $k-DNF$ expression which can be viewed as a "multi XOR" operation. More explicitly, we consider the $k-DNF$ expression to be true if at least one of its monomials is true, and at most m are true. We observe that the same arguments used in [3] to show that the $k-DNF$ is not learnable by $k-DNF$ still hold for the above case for any $m \geq 1$.

Theorem 7: The restricted version of the k–DNF in which at most m of the monomials can be true is learnable by a multilinear equation of degree bounded by $m \cdot k$.

Proof: We will consider only the case of monotone monomials. The case of non monotone monomials can be handled in the same method as in the proof of Theorem 5. Let the expression be $m_1 \vee m_2 \vee \cdots \vee m_N$, where $m_i = x_1^i \wedge \cdots \wedge x_{k_i}^i$. The following is a representation by an $m \cdot k$ degree multilinear function: $\prod_{\alpha=1}^{m}(\sum_{i=1}^{N}(\prod_{j=1}^{k_i} x_j^i) - \alpha) = 0$ □

7. Concluding remarks

We have shown that the representation of learnable knowledge by the same class of representations as those that are to be learned may be disadvantageous, as some concepts that are not learnable in this way are learnable by other representations. The class of pseudo boolean multilinear functions appears to provide a good representation for learnable knowledge, enabling a unified learning algorithm for many of the known classes of learnable concepts, as well as to others, not previously known to be learnable.

References

1. L. G. Valiant, "A theory of the learnable", *Communications of the Assoc. for Computing Machinery* 27, 11 (1984), 1134-1142.
2. A. Blumer, A. Ehrenfeucht, D. Haussler and M. Warmuth, "Classifying learnable geometric concepts with the Vapnic-Chervonenkis dimension", *Eighteenth annual Assoc. for Computing Machinery symposium on theory of computing*, Berkely, California, May 1986.
3. B. K. Natarajan, "On learning boolean functions", *Proceedings of the nineteenth annual Assoc. for Computing Machinery symposium on theory of computing*, 1987, 296-304.
4. M. Kearns, M. Li, L. Pitt and L. G. Valiant, "On the learnability of boolean formulae", *Proceedings of the nineteenth annual Assoc. for Computing Machinery symposium on theory of computing*, 1987, 285-295.
5. M. Kearns, M. Li, L. Pitt and L. G. Valiant, "Recent results on boolean concept learning", *Proc. 4'th Intl. workshop on Machine Learning*, 1987.
6. A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, (second edition), McGraw-Hill, New York, 1984.