Information and Computation **152**, 188–204 (1999)
Article ID inco.1998.2785, available online at http://www.idealibrary.com on IDEAL®

Proper Learning Algorithm for Functions of k

Yoshifumi Sakai*

Department of Information and Computer Sciences, Faculty of Engineering, Toyo University, 2100 Kujirai, Kawagoe 350-8585, Japan
E-mail: sakai@cs.toyo.ac.jp

and

Eiji Takimoto and Akira Maruoka

Graduate School of Information Sciences, Tohoku University, Sendai 980-8579, Japan
E-mail: t2@ecei.tohoku.ac.jp, maruoka@ecei.tohoku.ac.jp

In this paper, we introduce a probabilistic distribution, called a smooth distribution, which is a generalization of variants of the uniform distribution such as q-bounded distribution and product distribution. Then, we give an algorithm that, under the smooth distribution, properly learns the class of functions of k terms given as $\mathscr{F}_k \circ \mathscr{T}_n{}^k = \{ g(f_1(v), ..., f_k(v)) \mid g \in \mathscr{F}_k, f_1, ..., f_k \in \mathscr{T}_n \}$ in polynomial time for constant k, where $\mathscr{F}_k$ is the class of all Boolean functions of k variables and $\mathscr{T}_n$ is the class of terms over n variables. Although class $\mathscr{F}_k \circ \mathscr{T}_n{}^k$ was shown by Blum and Singh to be learned using DNF as the hypothesis class, it has remained open whether it is properly learnable under a distribution-free setting. © 1999 Academic Press

1. INTRODUCTION

Since Valiant introduced the PAC learning model (Valiant, 1984), much effort has been devoted to characterize learnable classes of concepts in this model. Among such classes are the ones represented by some restricted Boolean formulas such as DNF, CNF, k-DNF, k-CNF, k-term DNF, and k-clause CNF, as well as the ones, such as threshold functions, given by describing Boolean functions in terms of positive vectors for the functions. In this paper, as opposed to learning these simple functions, we explore how to learn functions expressed as functions of simple functions, that is, compositions $g \circ f$ of two simple functions g and f. It turns out that the difficulty of learning such a composition $g \circ f$ without knowing the output of the function f is overcome by means of monotone-based expansion of the function g, which is introduced in this paper. More specifically, let $\mathscr{G}$ and $\mathscr{F}$ denote classes of k variable Boolean functions and n variable Boolean functions, respectively.

* Corresponding author.

A target function to be learned is written as $g \circ (f_1, ..., f_k)$, where $g \in \mathscr{G}$, $f_1, ..., f_k \in \mathscr{F}$, and $\circ$ denotes the composition of functions. The target function $g \circ (f_1, ..., f_k)$ takes the value $g(f_1(v), ..., f_k(v))$ for a vector v in $\{0, 1\}^n$. To learn a target function given as $g \circ (f_1, ..., f_k)$, a learning algorithm is assumed to use examples of the form $(v, g(f_1(v), ..., f_k(v)))$, but it cannot use examples of the form $(v, f_i(v))$ for $1 \leqslant i \leqslant k$. So, even if both $\mathscr{G}$ and $\mathscr{F}$ are learnable, it is hard to learn $\mathscr{G} \circ \mathscr{F}^k$ in general. For example, let $\mathscr{G}_{\mathrm{OR}}$ consist of an OR function of k variables, and let $\mathscr{T}_n$ be the class of all monomials of n variables. In the distribution-free setting, it is known that $\mathscr{G}_{\mathrm{OR}} \circ \mathscr{T}_n{}^k$, denoted usually k-term DNF, is not properly learnable unless RP = NP (Kearns *et al.*, 1987; Pitt and Valiant 1988), whereas both $\mathscr{G}_{\mathrm{OR}}$ and $\mathscr{T}_n$ are learnable.

Blum and Singh (1990) studied the learnability of the class $\mathscr{G}_k \circ \mathscr{T}_n{}^k$, denoted usually $\mathscr{F}_{k\text{-term}}$, where $\mathscr{G}_k$ denotes the class of all Boolean functions of k variables, and they showed that, for constant k, $\mathscr{F}_{k\text{-term}}$ is learnable by hypothesis class $O(n^{k+1})$-term DNF in the distribution-free setting. Furthermore, they showed that, for any symmetric function g other than AND, NAND, TRUE, and FALSE, it is NP-hard to learn $\{g\} \circ \mathscr{T}_n^k$ properly.

In this paper, we first introduce a class of probabilistic distributions, called smooth distributions, which is a generalization of the uniform distribution and includes both q-bounded distributions (Flammini *et al.*, 1992) and product distributions (Kucera *et al.*, 1994) which have been dealt with in literature as variants of the uniform distribution. The main result of this paper says that there exists an algorithm that properly learns $\mathscr{F}_{k\text{-term}}$ under the smooth distribution in polynomial time for constant k.

Before proceeding to the next section, it would be helpful to give an intuitive idea behind our algorithm which is based on two lemmas. These two lemmas will be referred to as the Expansion Lemma (Lemma 2) and the Selection Lemma (Lemma 4) in Section 3. For the purpose of explanation we take a target function f in $\mathscr{F}_{k\text{-term}}$ given as $f = g(t_1, t_2, t_3, t_4)$, where $g = y_1 y_3 \vee y_2 y_4 \vee y_1 \bar{y}_4$, and $t_1 = x_1$, $t_2 = x_1 x_2 \bar{x}_3$, $t_3 = x_3 \bar{x}_4$, $t_4 = \bar{x}_5$. Note that to get the target function we may substitute x_1, $x_1 x_2 \bar{x}_3$, $x_3 \bar{x}_4$, $\bar{x}_5$ to y_1, y_2, y_3, y_4, respectively. If we put, as illustrated in Fig. 1, $g_1 = y_1 \vee y_2 y_4$, $g_2 = y_1 y_4$, and $g_3 = y_1 y_2 y_4 \vee y_1 y_3 y_4$, it turns out that g is expressed as $g_1 \oplus g_2 \oplus g_3$. This is the crucial observation on which our algorithm relies. In fact, as the Expansion Lemma says, any function g can be expressed as $g_1 \oplus g_2 \oplus \cdots \oplus g_d$ for appropriately chosen monotone functions $g_1, ..., g_d$ that satisfy $g_1 > g_2 > \cdots > g_d$ under the usual inclusion relation $>$. In general, given monotone functions $g_1, g_2, ..., g_d$ such that $g_1 > g_2 > \cdots > g_d$, it is natural to consider layers L_i that consist of vectors w such that $g_1(w) = \cdots = g_i(w) = 1$ and $g_{i+1}(w) = \cdots = g_d(w) = 0$, where $0 \leqslant i \leqslant d$. Then the whole set $\{0, 1\}^k$ is partitioned into layers $L_0, L_1, ..., L_d$.

We shall actually take as the domain of g the set of vectors w that satisfy, as well as some technical ones, the condition that $(t_1(v), ..., t_k(v)) = w$ holds with not too small probability when v is drawn according to a variant of the uniform distribution, called a smooth distribution. But for the sake of simplicity, we consider $\{0, 1\}^k$ as the domain of g for the time being. In our case we have $L_0 = \{(0, 0, 0, 0), (0, 1, 0, 0), (0, 0, 1, 0), (0, 0, 0, 1), (0, 1, 1, 0), (0, 0, 1, 1)\}$, $L_1 = \{(1, 0, 0, 0), (1, 1, 0, 0), (1, 0, 1, 0), (0, 1, 0, 1), (1, 1, 1, 0), (0, 1, 1, 1)\}$, $L_2 = \{(1, 0, 0, 1)\}$, and

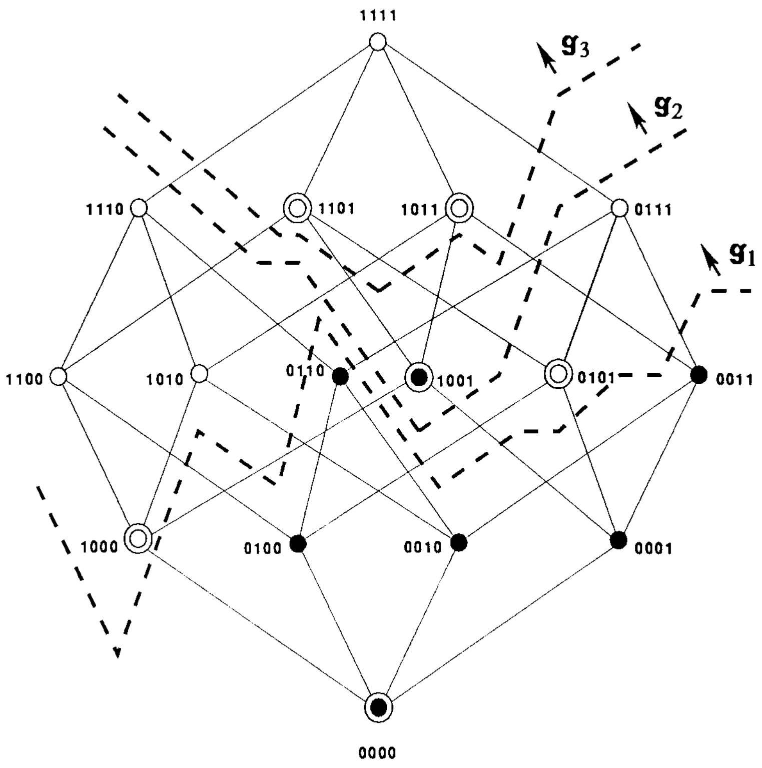

FIG. 1. The target function g is expressed as $g = g_1 \oplus g_2 \oplus g_3$, where $g_1 = y_1 \vee y_2 y_4$, $g_2 = y_1 y_4$, and $g_3 = y_1 y_2 y_4 \vee y_1 y_3 y_4$. Open circles in the Boolean cube represent the points where the function g takes value 1, whereas solid circles represent the points where g takes value 0. Double circles indicate that the corresponding points are the minimal points in the layers.

$L_3 = \{(1, 1, 0, 1), (1, 0, 1, 1), (1, 1, 1, 1)\}$. By definition it is clear to see that the function g changes its value when we go across the boundaries between layers: The function g takes value 1 for vectors in $L_1 \cup L_3$, while it takes value 0 for vectors in $L_0 \cup L_2$. Note that vectors in $L_1 \cup L_3$ are represented as open circles, whereas those in $L_0 \cup L_2$ are represented as solid circles. Furthermore, each layer is completely specified by the minimal vectors in the layer. Denoting by $\mathrm{Min}(L_i)$ the set of such minimal vectors in layer L_i, we have $\mathrm{Min}(L_1) = \{(1, 0, 0, 0), (0, 1, 0, 1)\}$, $\mathrm{Min}(L_2) = \{(1, 0, 0, 1)\}$, and $\mathrm{Min}(L_3) = \{(1, 1, 0, 1), (1, 0, 1, 1)\}$. Let M_g denote the collection of all the minimal vectors; that is, $M_g = \mathrm{Min}(L_1) \cup \mathrm{Min}(L_2) \cup \mathrm{Min}(L_3)$. In Fig. 1, the minimal vectors are represented as double circles.

What makes it difficult to learn the target function is that our learning algorithm cannot observe the vector $(t_1(v), t_2(v), ..., t_k(v))$ for examples v, where $(t_1(v), t_2(v), ..., t_k(v)) \in \{0, 1\}^k$. All the information the algorithm can make use of is whether the examples are positive or negative. In other words, instead of the vector $(t_1(v), t_2(v), ..., t_k(v))$ the algorithm can only have the value $g(t_1(v), t_2(v), ..., t_k(v))$, for example v. So, we give up searching for $T = (t_1, ..., t_k)$. Instead we try to search for terms $\tilde{t}_1, ..., \tilde{t}_k$ such that $g \circ (\tilde{t}_1, ..., \tilde{t}_k)$ behaves as the target function $g \circ (t_1, ..., t_k)$ does. To do so, for each minimal vector w of the layers we try to get the term that is the conjunction of the t_i's such that $w^{(i)} = 1$, where $w^{(i)}$ denotes the ith component of w. Such a term will be denoted by $\tau_w(T)$. For example, we have $\tau_w(T) = x_1 x_2 \bar{x}_3 \bar{x}_5$ for $w = (1, 1, 0, 1)$ and $T = (x_1, x_1 x_2 \bar{x}_3, x_3 \bar{x}_4, \bar{x}_5)$. To infer terms $\tau_w(T)$ for the minimal vectors w is the most crucial part of our algorithm. Such terms $\tau_w(T)$ will be computed from bottom to top in the Boolean cube. Suppose that we have computed $\tau_{(1,0,0,0)}(T)$ $(= x_1)$ for the minimal vector $(1, 0, 0, 0)$ in L_1 and we are about to compute $\tau_{(1,0,0,1)}(T)$ $(= x_1 \bar{x}_5)$ for the minimal vector $(1, 0, 0, 1)$ in layer L_2 which corresponds to the negative vectors. Taking the conditions (by exhaustive search) that make sure $t_2(v) = 0$ and $t_3(v) = 0$, say, $x_2(v) = 0$ and $\bar{x}_4(v) = 0$, we collect a sufficiently large number of negative examples v such that $\tau_{(1,0,0,0)}(T)(v) = 1$, $x_2(v) = 0$, and $\bar{x}_4(v) = 0$, namely, negative examples v such that $v^{(1)} = 1$, $v^{(2)} = 0$, and $v^{(4)} = 1$. It is easy to see that, since vector v is a negative vector, $(t_1(v), t_2(v), t_3(v), t_4(v))$ belongs to L_2, provided that $v^{(1)} = 1$, $v^{(2)} = 0$, and $v^{(4)} = 1$ (hence, $t_1(v) = 1$, $t_2(v) = 0$, and $t_3(v) = 0$) hold. We therefore have that $t_4(v) = 1$. Lemma 7 in Section 3 gives the statement that is obtained by generalizing the arguments above. In order to get $\tau_{(1,0,0,1)}(T)$, all we have to do is to observe components that take the same value of all of these negative examples v such that $v^{(1)} = 1$, $v^{(2)} = 0$, $v^{(4)} = 1$ and to discard the components $v^{(2)}$ and $v^{(4)}$ which are used to make the irrelevant terms take the value 0. If the number of such negative examples is large enough, we can conclude that $\tau_{(1,0,0,1)}(T) = x_1 \bar{x}_5$ by the standard argument.

We now proceed to explaining how to get an hypothesis by using the terms obtained this way which are written as $\tau_w(T)$ for w in M_g. As mentioned above, our algorithm tries to obtain terms $\tilde{t}_1, ..., \tilde{t}_k$ such that $g \circ (\tilde{t}_1, ..., \tilde{t}_k)$ behaves like $g \circ (t_1, ..., t_k)$. The Selection Lemma in Section 3 shows how to compute such $\tilde{t}_1, ..., \tilde{t}_k$ from terms $\tau_w(T)$'s. The lemma simply says that, for $1 \leqslant i \leqslant k$, the term $\tilde{t}_i$ consists of the literals that appear in all the $\tau_w(T)$'s such that $w \in M_g$ and $w^{(i)} = 1$.

Actually, in our algorithm we compute a collection of candidates for such terms which includes all of the terms $\tilde{t}_1, ..., \tilde{t}_k$. Once such candidates are obtained, all we have to do is to search exhaustively for $\tilde{t}_1, ..., \tilde{t}_k$ and $\tilde{g}$ from the candidates and from all of the k variable functions, respectively, until the hypothesis $\tilde{g} \circ (\tilde{t}_1, ..., \tilde{t}_k)$ approximates the target function $g \circ (t_1, ..., t_k)$ with sufficient accuracy. Note that, since the parameter k is assumed to be constant, we can search for $\tilde{g}$ by exhaustive search.

2. PRELIMINARIES

In this paper we follow the standard terminologies in the PAC learning model unless otherwise stated. Let D be a probability distribution on Σ^n, where Σ denotes $\{0, 1\}$. For a vector v in Σ^n, $D(v)$ denotes the probability assigned to v and, for $V \subseteq \Sigma^n$, $D(V)$ denotes $\sum_{v \in V} D(v)$. A target function, denoted f, and an hypothesis, denoted h, are assumed to be Boolean functions of n variables $x_1, ..., x_n$. A Boolean function f is thought of as representing the set of vectors v as well, such that $f(v) = 1$. So $D(f)$ represents $D(\{v \mid f(v) = 1\})$ and $f \subseteq f'$ means $\{v \mid f(v) = 1\} \subseteq \{v \mid f'(v) = 1\}$. Furthermore, we adopt the notation $D(f \mid f')$ that denotes $D(f \wedge f')/D(f')$. A function h is said to ε-approximate f under D if and only if $D(f \oplus g) < \varepsilon$ holds. Let f be a function of n variables and let D be a distribution on

Σ^n. If $D(f)$ $(=D(\{v \mid f(v)=1\}))$ $\neq 0$, then D_f^+ is defined as $D_f^+(v)=D(v)/D(f)$ for v in f. Similarly, if $D(\bar{f}) \neq 0$, then D_f^- is defined as $D_f^-(v)=D(v)/D(\bar{f})$ for v in $\bar{f}$. When f is clear from the context, D_f^+ and D_f^- will be denoted simply by D^+ and D^-, respectively. The size of a Boolean function f is the number of symbols appearing in the shortest description of f under some reasonable encoding. Given a class of Boolean functions $\mathscr{F}$, $\mathscr{F}_{n,s}$ denotes the set of Boolean functions in $\mathscr{F}$ that have n variables and size at most s. In the following, we often identify a Boolean formula with the Boolean function that it represents.

Definition 1. Let $\mathscr{F}$ be a class of Boolean functions, and let $\mathscr{D}$ be a class of distributions. An algorithm L learns $\mathscr{F}$ under $\mathscr{D}$ if and only if for any positive integers n, s, any target function f in $\mathscr{F}_{n,s}$, any real numbers ε, δ with $0<\varepsilon$, $\delta<1$, and any distribution D on Σ^n in $\mathscr{D}$, when L is given as input n, s, ε, and δ as well as access to POS( ) and NEG( ) that generate positive and negative examples independently according to D^+ and D^-, respectively, L halts in time at most some polynomial in n, s, $1/\varepsilon$, and $1/\delta$, and outputs, with probability at least $1-\delta$, an hypothesis h in $\mathscr{F}_n$ that ε-approximates f under D. Furthermore, if there exists a learning algorithm for $\mathscr{F}$ under $\mathscr{D}$, then $\mathscr{F}$ is said to be learnable under $\mathscr{D}$.

For a vector v in Σ^n and an integer $1 \leqslant i \leqslant n$, let $v^{(i)}$ denote the ith component of v. For a vector v, let true(v) and false(v) denote $\{i \mid v^{(i)}=1\}$ and $\{i \mid v^{(i)}=0\}$, respectively. Let 0^n and 1^n denote vectors $(0, 0, ..., 0)$ and $(1, 1, ..., 1)$ in Σ^n, respectively. For v and v' in Σ^n, let $v \leqslant v'$ mean that $v^{(i)} \leqslant (v')^{(i)}$ for any $1 \leqslant i \leqslant n$, and let $v < v'$ mean that $v \leqslant v'$ and $v \neq v'$. For any subset V of Σ^n, let Min(V) denote the set of the minimal vectors in V, that is,

$$\mathrm{Min}(V)=\{v \in V \mid \forall v' \in V\; v' \not< v\},$$

and let mon(V) denote the monotone Boolean function of n variables defined as

$$\mathrm{mon}(V)(v)=\begin{cases} 1, & \exists v' \in V, \quad v' \leqslant v; \\ 0, & \text{otherwise.} \end{cases}$$

Note that mon$(V)=$ mon(Min(V)) for any subset V of Σ^n. Let X_n denote the set of Boolean variables $x_1, ..., x_n$, and let $\neg X_n$ denote the set $\{\bar{x}_i \mid x_i \in X_n\}$. Let $\mathscr{G}_n$ denote the set of all Boolean functions of variables in X_n. TRUE and FALSE denote constant functions that take identically 1 and 0, respectively. A conjunction of literals is called a term or a monomial. For a term t, lit(t) denotes the set of literals that appear in t. A literal y is said to be irrelevant with a term t if $y \notin \mathrm{lit}(t)$ and $\bar{y} \notin \mathrm{lit}(t)$, where $\bar{y}$ means x for $y=\bar{x}$. Let $\mathscr{T}_n$ denote the set of all terms of literals in $X_n \cup \neg X_n$. For a positive integer k, $\mathscr{T}_{n,\leqslant k}$ denotes the set of t's in $\mathscr{T}_n$ such that $|\mathrm{lit}(t)| \leqslant k$. For any vector v in Σ^n, τ_v denotes the term of n variables defined as

$$\tau_v = \bigwedge_{i \in \mathrm{true}(v)} x_i.$$

We adopt the convention that $\tau_{0^n}=$ TRUE.

For a Boolean function g of k variables and k-tuple $T=(t_1, ..., t_k)$ of terms, $g(T)$ denotes the Boolean function of n variables that takes value $g(t_1(v), ..., t_k(v))$ for a vector v in Σ^n. A Boolean function that can be represented as $g(T)$ for some g in $\mathscr{G}_k$ and for some $T=(t_1, ..., t_k)$ in $\mathscr{T}_n^k$ is called a function of k terms, and the class of functions of k terms is denoted by $\mathscr{F}_{k\text{-term}}$. For example, the class $\mathscr{F}_{2\text{-term}}$ includes the function $(x_1 \wedge \bar{x}_2) \oplus (\bar{x}_1 \wedge x_3 \wedge x_4)$, where $\oplus$ denotes the exclusive OR function. Since T is considered to be a function from Σ^n to Σ^k, a function $g(T)$ in $\mathscr{F}_{k\text{-term}}$ can be thought of as the composition of g and T which will also be denoted by $g \circ T$. Similarly, in the following, we use notations such as $\tau_v(T)$ and $\tau_v \circ T$.

In this paper we introduce a variant of the uniform distribution, called a smooth distribution, which is a generalization of all the distributions, such as q-bounded and product distributions (Flammini *et al.*, 1992; Kucera *et al.*, 1994) that have been dealt with in literature so far as variants of the uniform distribution.

DEFINITION 2. For a positive integer n and a real number $0<p\leqslant 1$, a probability distribution D on Σ^n is p-smooth if, for any vectors v and v' in Σ^n with Hamming distance 1, $D(v)/D(v')\geqslant p$ holds.

Before closing this section, we give a lemma due to Chernoff that will be used in the following sections.

LEMMA 1 (Kearns *et al.*, 1987). *For any real number $0\leqslant p\leqslant 1$, positive integer m and real number $0\leqslant b\leqslant 1$, let $LE(p, m, b)$ denote the probability of having at most $(1-b)\,mp$ successes in m independent trials with probability of success at least p, and let $GE(p, m, b)$ denote the probability of having at least $(1+b)\,mp$ successes in m independent trials with probability of success at most p. Then we have*

$$LE(p, m, b)\leqslant e^{-b^2mp/2},$$

$$GE(p, m, b)\leqslant e^{-b^2mp/3}.$$

3. LEARNING ALGORITHM

Before proceeding to our learning algorithm for functions of k terms, we give two lemmas, called the Expansion Lemma and the Selection Lemma, on which the learning algorithm is based. Let g be an arbitrary Boolean function of k variables which is defined on a subset, denoted W, of Σ^k. The domain W will be fixed later appropriately so that our algorithm works well. The expansion lemma simply says that any Boolean function of k variables on domain W can be expanded as the XOR of at most $k+1$ monotone Boolean functions on W.

LEMMA 2 (Expansion Lemma). *Let g be any Boolean function of k variables, whose domain is given by $W\subseteq\Sigma^k$. Define the sequences of Boolean functions g_i and h_i defined on W for $1\leqslant i\leqslant k$ inductively as follows. Let $h_0=g$ and for $1\leqslant i\leqslant k$, define*

$$g_i=\text{mon}(h_{i-1})$$

and

$$h_i = \text{mon}(h_{i-1}) \wedge \bar{h}_{i-1}.$$

Let d be the least integer i such that $h_i = \text{FALSE}$. Then, $g = g_1 \oplus \cdots \oplus g_d$, $g_1 > \cdots > g_d$, and $d \leqslant k+1$ hold.

Proof. Since the general case is verified in a similar way, we assume that $W = \Sigma^k$. First we show that $g = \bigoplus_{1 \leqslant i \leqslant d} g_i$. Since $\text{mon}(h_{i-1}) \geqslant h_{i-1}$, we have $\overline{\text{mon}(h_{i-1})} \wedge h_{i-1} = \text{FALSE}$. So h_i can be equivalently represented as $h_i = \text{mon}(h_{i-1}) \oplus h_{i-1} = g_i \oplus h_{i-1}$. Applying these equalities inductively, we have $h_i = g \oplus g_1 \oplus \cdots \oplus g_i$. Since $h_d = \text{FALSE}$, we have $g \oplus \bigoplus_{1 \leqslant i \leqslant d} g_i = \text{FALSE}$, or equivalently $g = \bigoplus_{1 \leqslant i \leqslant d} g_i$.

Next we show that $g_i > g_{i+1}$ for $1 \leqslant i < d$. Fix $1 \leqslant i < d$ arbitrarily. Since $g_{i+1} = \text{mon}(h_i) = \text{mon}(\text{mon}(h_{i-1}) \wedge \bar{h}_{i-1}) \leqslant \text{mon}(h_{i-1}) \wedge \text{mon}(\bar{h}_{i-1}) = g_i \wedge \text{mon}(\bar{h}_{i-1})$, we have $g_i \geqslant g_{i+1}$. On the other hand, since $h_i = \text{mon}(h_{i-1}) \wedge \bar{h}_{i-1} \neq \text{FALSE}$, there exists w such that $\text{mon}(h_{i-1})(w) = 1$ and $h_{i-1}(w) = 0$. Let w be the minimal among such vectors. Then there exists w' such that $h_{i-1}(w') = 1$ and $w' < w$. Clearly $g_i(w') = \text{mon}(h_{i-1})(w') = 1$. On the other hand, since w is the minimal over the vector that makes both $\text{mon}(h_{i-1})$ and $\bar{h}_{i-1}$ take value 1, that is, that makes h_i take value 1, it follows that $g_{i+1}(w') = \text{mon}(h_i)(w') = 0$ in view of $w' < w$. Thus, together with $g_i \geqslant g_{i+1}$, we have $g_i > g_{i+1}$.

Finally we verify that $d \leqslant k+1$ by showing that there exist d vectors $v_0 < v_1 < \cdots < v_{d-1}$, which immediately implies $d \leqslant k+1$. Since $h_{d-1} \neq \text{FALSE}$ we can choose a vector v_{d-1} such that $h_{d-1}(v_{d-1}) = 1$. Then, since $h_i = \text{mon}(h_{i-1}) \wedge \bar{h}_{i-1}$, there exists a vector $v_{i-1} < v_i$ such that $h_{i-1}(v_{i-1}) = 0$ for $1 \leqslant i \leqslant d-1$. ∎

In what follows we simply refer to an XOR-expansion based on monotone DNF formulas as an $\oplus$MDNF formula. From the Expansion Lemma it is easy to derive the uniqueness of the $\oplus$MDNF expansion. In what follows, the domain $W \subseteq \Sigma^k$ is assumed to contain the vector 0^k. Furthermore, g, g_i, and h_i are supposed to be functions defined on the domain W.

Let $g = \bigoplus_{1 \leqslant i \leqslant d} g_i$ be an $\oplus$MDNF formula. Then, for any w in $W \subseteq \Sigma^k$, there exists an $0 \leqslant i \leqslant d$ such that $g_1(w) = \cdots = g_i(w) = 1$ and $g_{i+1}(w) = \cdots = g_d(w) = 0$ because $g_1 > \cdots > g_d$. Note that the value $g(w)$ is determined by the parity of such i. The collection of such vectors is called the ith layer of g, which is denoted by $L_{g,i}$. That is, $L_{g,i} = g_i - g_{i+1}$ for $1 \leqslant i \leqslant d$ and $L_{g,0} = W - g_1$, where g_{d+1} is assumed to be FALSE, which takes value 0 for any vector. When no confusion arises, $L_{g,i}$ is simply written as L_i. It is clear that the domain $W \subseteq \Sigma^k$ can be partitioned into $d+1$ layers $L_0, ..., L_d$ so that g takes the same value, which is given by the parity of i, in each layer and the opposite value in any neighboring layers. It is also clear that the set $\text{Min}(L_i)$ consists of the vectors w such that τ_w is a prime implicant of g_i. So, using the minimal vectors in L_i, the function g_i is written as

$$g_i = \bigvee_{w \in \text{Min}(L_i)} \tau_w.$$

Since g's domain W is assumed to contain the vector 0^k, $\mathrm{Min}(L_0)=\{0^k\}$ if $g(0^k)=0$ and $L_0=\emptyset$ otherwise. In the latter case, $\mathrm{Min}(L_1)=\{0^k\}$ and $g_1=\mathrm{TRUE}$.

Let U be a set, and $A_1, ..., A_k$ be subsets of U. Let M be a subset of Σ^k. For w in Σ^k, let $\bigcup_w (A_1, ..., A_k)$ denote subset $\bigcup \{A_i \mid w^{(i)}=1\}$, namely, the union of subsets A_i such that the ith component of vector w is 1. The problem we consider is how to construct a collection of subsets $\{B_1, ..., B_k\}$ such that $\bigcup_w (A_1, ..., A_k)=\bigcup_w (B_1, ..., B_k)$ holds for any w in M, provided that all the information we get is subsets written as $\bigcup_w (A_1, ..., A_k)$ for any w in M. Note that, if all of the k unit vectors belong to M, then the problem becomes trivial, because for $1 \leqslant i \leqslant k$, we can put $B_i=\bigcup_{e_i} (A_1, ..., A_k)$ $(=A_i)$, where e_i is the unit vector with the ith component being 1. The next Selection Lemma gives us an answer to the problem for M given arbitrarily.

Lemma 3 (Selection Lemma). *Let U be a set and $A_1, ..., A_k$ be subsets of U. Let M be a subset of Σ^k. Put*

$$B_i=\bigcap \left\{ \bigcup_{w'} (A_1, ..., A_k) \,\middle|\, w' \in M \text{ and } (w')^{(i)}=1 \right\}$$

for $1 \leqslant i \leqslant k$:

(i) *For any* $1 \leqslant i \leqslant k$,

$$A_i \subseteq B_i.$$

(ii) *For any w in M,*

$$\bigcup_w (A_1, ..., A_k)=\bigcup_w (B_1, ..., B_k).$$

Proof. By the definition of B_i, it is easy to see that $A_i \subseteq B_i$ holds for any $1 \leqslant i \leqslant k$, completing the proof of (i).

For the proof of (ii), since $\bigcup_w (A_1, ..., A_k)=\bigcup_w (B_1, ..., B_k)=\emptyset$ for $w=(0, ..., 0)$, we assume $w \neq (0, ..., 0)$. By (i), we have $\bigcup_w (A_1, ..., A_k) \subseteq \bigcup_w (B_1, ..., B_k)$ for any w in M. On the other hand, for any w in M and any $1 \leqslant i \leqslant k$ such that $w^{(i)}=1$, we have $B_i=\bigcap \{\bigcup_{w'} (A_1, ..., A_k) \mid w' \in M$ and $(w')^{(i)}=1\} \subseteq \bigcup_w (A_1, ..., A_k)$, which implies $\bigcup_w (B_1, ..., B_k) \subseteq \bigcup_w (A_1, ..., A_k)$. This completes the proof of (ii). ▮

Let $g \circ T$ denote a target function we want to learn. Putting $T=(t_1, ..., t_k)$, the value of T for an example v is given as $T(v)=(t_1(v), ..., t_k(v))$. Instead of seeking for $T=(t_1, ..., t_k)$, we will compute $\tilde{T}=(\tilde{t}_1, ..., \tilde{t}_k)$ such that $T(v)$ and $\tilde{T}(v)$ take the same value with large probability when an example v is taken from Σ^n according to a smooth probability distribution D. So $(g \circ T)(v)$ and $(g \circ \tilde{T})(v)$ take the same value with large probability.

Let

$$W'=\{w \in T(\Sigma^n) \mid D(\{v \mid T(v)=w\}) \geqslant \varepsilon/2^{k+1}\},$$

where $T(\Sigma^n)$ denotes the set $\{w \in \Sigma^k \mid \exists v \in \Sigma^n\ T(v) = w\}$. Furthermore, W is defined to be the minimum downward closed set of W', that is,

$$W = \{w \in T(\Sigma^n) \mid \exists w' \in W'\ w \leqslant w'\}.$$

We will see later that the condition that W is downward closed is assumed just for some technical reason. In what follows we take W as the domain of the function g. That is, we consider g as the function from W to $\{0, 1\}$ unless stated otherwise. Let the function g with the domain W have the $\bigoplus$MDNF expansion $g = \bigoplus_{1 \leqslant i \leqslant d} g_i$. Furthermore, let $M_g = \bigcup_{1 \leqslant i \leqslant d} \mathrm{Min}(L_i)$, where L_i denotes the ith layer of the function g. In the next lemma, the term made up of literals in $\bigcap \{\mathrm{lit}(\tau_w(T)) \mid w \in M_g \text{ and } w^{(i)} = 1\}$ is written as $\bigcap \{\tau_w(T) \mid w \in M_g \text{ and } w^{(i)} = 1\}$ for simplicity. We adopt the convention that the term corresponding to the empty set denotes FALSE. Taking M_g and $\mathrm{lit}(t_i)$ as M and A_i in the Selection Lemma, the lemma can be restated as follows.

LEMMA 4 (Selection Lemma). *Let $T = (t_1, ..., t_k)$ and M_g be defined as $M_g = \bigcup_{1 \leqslant i \leqslant d} \mathrm{Min}(L_i)$. Put*

$$\tilde{t}_i = \bigcap \{\tau_{w'}(T) \mid w' \in M_g \text{ and } (w')^{(i)} = 1\}$$

for $1 \leqslant i \leqslant k$ and

$$\tilde{T} = (\tilde{t}_1, ..., \tilde{t}_k):$$

(i) *For any $1 \leqslant i \leqslant k$,*

$$\mathrm{lit}(t_i) \subseteq \mathrm{lit}(\tilde{t}_i).$$

(ii) *For any w in M_g,*

$$\tau_w(T) = \tau_w(\tilde{T}).$$

We are now ready to explain the rough idea of our algorithm which is given in Fig. 2. Put $g = y_1 y_3 \vee y_2 y_4 \vee y_1 \bar{y}_4$ as in the previous example and let $T = (t_1, t_2, t_3, t_4)$ be such that $t_1 = x_1$, $t_2 = x_1 x_2 \bar{x}_3$, $t_3 = x_3 \bar{x}_4$, and $t_4 = \bar{x}_5$. Putting g and T in this way, we take $g(T)$ as the target function. It was shown in the previous example that $M_g = \bigcup_{i=1}^4 \mathrm{Min}(L_i) = \{(1, 0, 0, 0), (0, 1, 0, 1), (1, 0, 0, 1), (1, 1, 0, 1), (1, 0, 1, 1)\}$. Then by applying Lemma 4 to terms t_1, t_2, t_3, t_4, and M_g, which is not known to the algorithm, we have, for instance,

$$\begin{aligned}\mathrm{lit}(\tilde{t}_4) &= \bigcap \{\tau_{(0,1,0,1)}, \tau_{(1,0,0,1)}, \tau_{(1,1,0,1)}, \tau_{(1,0,1,1)}\} \\ &= \{x_1, x_2, \bar{x}_3, \bar{x}_5\} \cap \{x_1, \bar{x}_5\} \cap \{x_1, x_2, \bar{x}_3, \bar{x}_5\} \cap \{x_1, x_3, \bar{x}_4, \bar{x}_5\} \\ &= \{x_1, \bar{x}_5\};\end{aligned}$$

hence $\tilde{t}_4 = x_1 \bar{x}_5$. In a similar way, we have $\tilde{t}_1 = x_1$, $\tilde{t}_2 = x_1 x_2 \bar{x}_3 \bar{x}_5$, $\tilde{t}_3 = x_1 x_3 \bar{x}_4 \bar{x}_5$.

Our algorithm computes $\tau_w(T)$ repeatedly for each minimal vector w belonging to layer L_1 up to L_4. Suppose we have computed $\tau_{w'}(T)$ for each w' in $\mathrm{Min}(L_1)$. Based on such $\tau_{w'}(T)$ the algorithm computes $\tau_w(T)$ for w in $\mathrm{Min}(L_2) = \{(1, 0, 0, 1)\}$. To do so the algorithm takes all the possible pairs (s, r) of terms s in $\{\tau_{w'}(T) \mid w' \in \mathrm{Min}(L_1)\}$ and terms r consisting of at most k literals. Suppose that, as such a combination, term $s = \tau_{(1,0,0,0)}(T) = x_1$ and term $r = \bar{x}_2 x_4$ are chosen. Since $g((1, 0, 0, 1)) = 0$, negative vectors v's that satisfy both $\tau_{(1,0,0,0)}(T)(v) = 1$ (or, equivalently, $v^{(1)} = 1$) and $\bar{x}_2 x_4(v) = 1$ (or, equivalently, $v^{(2)} = 0$ and $v^{(4)} = 1$) are collected to form a collection V of such vectors v's. It is easy to see that $\tau_{(1,0,0,0)}(T)(v) = 1$ is equivalent to $T(v) \geqslant (1, 0, 0, 0)$, while $\tau_{(1,0,0,0)}(T)(v) = 1$, $v^{(2)} = 0$, $v^{(4)} = 1$ and $g(T(v)) = 0$ (or, equivalently, vector v is negative) implies that $T(v) = (1, 0, 0, 1)$. Since $T(v) = (1, 0, 0, 1)$ is equivalent to $x_1(v) = 1$, $x_1 x_2 \bar{x}_3(v) = 0$, $x_3 \bar{x}_4(v) = 0$, and $\bar{x}_5(v) = 1$, the term $\tau_{(1,0,0,1)}(T) = x_1 \bar{x}_5$ is obtained by taking bit-wise AND of all the vectors in the sufficiently large collection V. By repeating what is described above, we obtain $\tau_w(T)$ for all w in M_g. In fact, the collection of such terms that are described as $\tau_w(T)$ for some w in L_l is denoted by $\mathcal{U}_l$ in Fig. 2. Notice that we

```
Algorithm LEARN($n, \varepsilon, \delta$):        (* $\beta = \varepsilon p^k / 2^{2k-1}$ *)
begin
  $m \leftarrow \max\left\{\frac{32}{\beta}, \frac{8}{3\beta p}, \frac{32}{\varepsilon}\right\} \ln \frac{\max\{2^{k+3}(k+1)n, (2n)^{2^{k+2}} k^3\}}{\delta}$;
  obtain sets $S^+$ and $S^-$ of $m$ positive examples and $m$ negative examples
    drawn according to $D^+$ and $D^-$, respectively;
  $\mathcal{U}_0 \leftarrow \{\mathrm{TRUE}\}$;
  $\mathcal{U}_1, \ldots, \mathcal{U}_{k+1} \leftarrow \emptyset$;
  for $l \leftarrow 1$ step 1 until $k+1$ do
    for each $(s, r) \in \mathcal{U}_{l-1} \times \mathcal{T}_{n,\leq k}$ do
      begin
        $V \leftarrow \{v \in S^{parity(l)} \mid (s \wedge r)(v) = 1\}$;        (* multiset *)
        if $|V| \geq \frac{3}{4}\beta m$ then
          begin
            $Y \leftarrow \{y \in X_n \cup \neg X_n \mid \forall v \in V \quad y(v) = 1\}$;
            $\mathcal{U}_l \leftarrow \mathcal{U}_l \cup \{\wedge(Y \backslash \rho) \mid \rho \subseteq lit(r)\}$
          end
      end;
  $\mathcal{U} \leftarrow \bigcup_{1 \leq l \leq k+1} \mathcal{U}_l$;
  $\hat{\mathcal{U}} \leftarrow \left\{\wedge\left(\bigcap_{u \in \mathcal{U}'} u\right) \middle| \mathcal{U}' \subseteq \mathcal{U}, |\mathcal{U}'| \leq 2^{k-1}\right\} \cup \{\mathrm{FALSE}\}$;
  $\mathcal{H} \leftarrow \{g'(T') \mid g' \in \mathcal{G}_k,\ T' = (t'_1, \ldots, t'_k) \in \hat{\mathcal{U}}^k\}$;
    (* $\mathcal{G}_k$ denotes the set of Boolean functions of $k$ variables *)
  for each $h \in \mathcal{H}$ do
    if $\left|\{v \in S^+ \mid h(v) = 0\}\right| < \frac{3}{4}\varepsilon m$ and $\left|\{v \in S^- \mid h(v) = 1\}\right| < \frac{3}{4}\varepsilon m$ then
      output $h$
end.
```

FIG. 2. Algorithm LEARN.

only obtain a term described as $\tau_w(T)$ for some w in M_g without knowing the corresponding vector w. So in order to obtain $\tilde{t}_i$ described as in Lemma 6 in terms of $\tau_{w'}(T)$'s for w' in M_g, we take, exhaustively, the intersection of possible collections of literals in an appropriate number of such terms. The collection, denoted $\hat{\mathscr{U}}$, of terms obtained in this way includes all of the desired terms $\tilde{t}_i$'s for $1 \leqslant i \leqslant k$. Once $\hat{\mathscr{U}}$ is computed, the algorithm computes $\tilde{g}(\tilde{T})$ by trying exhaustively all $\tilde{g}$'s of k variables and all k-tuples of terms in $\hat{\mathscr{U}}$ until a sufficiently accurate hypothesis $\tilde{g}(\tilde{T})$ is obtained.

The next lemma says that $g \circ \tilde{T}$ behaves the same as $g \circ T$ does on the domain $\{v \in \Sigma^k \mid T(v) \in W\}$.

LEMMA 5. *Let g be the function from Σ^k to Σ given by the formula*

$$\bigoplus_{1 \leqslant i \leqslant d} \bigvee \{\tau_w \mid w \in \mathrm{Min}(L_i)\}.$$

Then for any v in Σ^n such that $T(v) \in W$,

$$g(T(v)) = g(\tilde{T}(v)).$$

Proof. For an arbitrary vector v in Σ^n such that $T(v) \in W$, let $w = T(v)$. For simplicity of notation in the following argument, put $g_0 = \mathrm{TRUE}$ and $g_{d+1} = \mathrm{FALSE}$. Let $0 \leqslant j \leqslant d$ be the subscript such that $T(v) \in g_j - g_{j+1}$, that is, $g_j(T(v)) = 1$ and $g_{j+1}(T(v)) = 0$. Since g takes the same value for w's that belong to the same layer, it suffices to show that $\tilde{T}(v) \in g_j - g_{j+1}$.

Since W is downward closed, there exists the minimal vector w_{m} in g_j such that $w_{\mathrm{m}} \leqslant w$. Then since $\tau_{w_{\mathrm{m}}}(T) = \tau_{w_{\mathrm{m}}}(\tilde{T})$ by (ii) in Lemma 4, we have $\tau_{w_{\mathrm{m}}}(\tilde{T})(v) = \tau_{w_{\mathrm{m}}}(T)(v) = 1$, which implies $\tilde{T}(v) \geqslant w_{\mathrm{m}}$. Thus by the monotonicity of g_j and the fact that $g_j(w_{\mathrm{m}}) = 1$, we have $g_j(\tilde{T}(v)) = 1$.

On the other hand, by (i) in Lemma 4, $\tilde{T}(v) \leqslant T(v)$ holds. Thus, by the monotonicity of g_{j+1} and the fact that $g_{j+1}(T(v)) = 0$, we have $g_{j+1}(\tilde{T}(v)) = 0$. ∎

It is easy to see from Lemma 5 that the next lemma holds.

LEMMA 6. *$g \circ \tilde{T}$ $\varepsilon/2$-approximates $g \circ T$.*

As mentioned in the example above, our algorithm computes a collection of terms, denoted $\mathscr{U}_l$, such that $\{\tau_w(T) \mid w \in \mathrm{Min}(L_l)\} \subseteq \mathscr{U}_l$ from $l = 0$ up to d, repeatedly. More precisely, given $\tau_{w'}(T)$ for w' in $\mathrm{Min}\{w'' \in W \mid g_{l-1}(w'') = 1\}$ we seek for $\tau_w(T)$ for w in $\mathrm{Min}\{w'' \in W \mid g_l(w'') = 1\}$ such that $w' < w$. The relation between such vectors is described in terms of the notation defined as

$$ANCES(w') = \mathrm{Min}\{w \in W \mid w' < w \text{ and } g(w') \neq g(w)\}.$$

To compute such a term $\tau_w(T)$ for $\tau_{w'}(T)$ with $w \in ANCES(w')$ without knowing the value $T(v)$ for vectors v's, we guess a term r that kills all the unnecessary terms among $t_1, ..., t_k$, that is, terms t_i's such that $w^{(i)} = 0$, where $T = (t_1, ..., t_k)$. More precisely, the term r is taken so that, for all the unnecessary terms t_i's, r contains the negation $\bar{y}_i$ of some literal y_i in $\mathrm{lit}(t_i)$. The collection of such terms, denoted R_w, is defined as

$$R_w = \left\{ \bigwedge_{i \in \mathrm{false}(w)} \bar{y}_i \;\middle|\; y_i \in \mathrm{lit}(t_i) \text{ for } i \in \mathrm{false}(w) \right\}.$$

The algorithm computes $\tau_w(T)$ based on the next lemma.

LEMMA 7. *For any vector w' in M_g, any vector w in $ANCES(w')$, and any term r in R_w,*

$$\tau_w(T) \wedge r = \tau_{w'}(T) \wedge r \wedge (g \circ T)^{g(w)},$$

where $(g \circ T)^{g(w)}$ denotes $g \circ T$ if $g(w) = 1$ and $\overline{g \circ T}$ otherwise.

Proof. Let w', w, and r be as in the lemma. Then the statement of the lemma is restated as

$$\tau_w(T) \cap r = \tau_{w'}(T) \cap r \cap \{v' \in \Sigma^k \mid g(T(v')) = g(w)\},$$

where $\tau_w(T)$ denotes $\{v' \in \Sigma^k \mid \tau_w(T(v')) = 1\}$, and similarly for $\tau_{w'}(T)$ and r. It is easy to see by the definition of R_w and the fact that $r \in R_w$ that

$$v \in \tau_w(T) \cap r \Leftrightarrow T(v) = w, \qquad r(v) = 1.$$

On the other hand, we have

$$v \in \tau_{w'}(T) \cap r \Leftrightarrow w' \leqslant T(v) \leqslant w, \qquad r(v) = 1.$$

Since $g(T(v))$ for $w' \leqslant T(v) \leqslant w$ takes a different value from $g(w')$ only when $T(v) = w$, it is easy to see that

$$\begin{aligned} & v \in \tau_{w'}(T) \cap r \cap \{v' \in \Sigma^k \mid g(T(v')) = g(w)\} \\ \Leftrightarrow\ & v \in \tau_{w'}(T) \cap r, \qquad g(T(v)) = g(w) \\ \Leftrightarrow\ & T(v) = w, \qquad r(v) = 1. \end{aligned}$$

This completes the proof. ▌

4. CORRECTNESS

The next lemma is immediate from the definition of the smoothness.

LEMMA 8. *Let distribution D be p-smooth. Then for any terms t and t' such that $t \wedge t' \neq \mathrm{FALSE}$,*

$$D(t \mid t') \geqslant (p/2)^{|lit(t)|}.$$

As will be shown in the proof of Lemma 9, our algorithm exploits the properties of a smooth distribution in terms of the condition in Lemma 8.

LEMMA 9. *Let distribution D be p-smooth. For any w in M_g and r in R_w such that $\tau_w(T) \wedge r \neq \text{FALSE}$,*

$$D(\tau_w(T) \wedge r) \geqslant \beta$$

holds, where $\beta = \varepsilon p^k/2^{2k+1}$. Furthermore for any literal y that is irrelevant with $\tau_w(T) \wedge r$,

$$p/2 \leqslant D(y \mid \tau_w(T) \wedge r) \leqslant 1 - p/2$$

holds.

Proof. By the definition of W there exists w' in W such that $w \leqslant w'$ and $D(\{v \mid T(v) = w'\}) \geqslant \varepsilon/2^{k+1}$, which implies that $D(\tau_w(T)) = D(\{v \mid T(v) \geqslant w\}) \geqslant D(\{v \mid T(v) = w'\}) \geqslant \varepsilon/2^{k+1}$. By Lemma 8, we have $D(r \mid \tau_w(T)) \geqslant (p/2)^{|lit(r)|} \geqslant (p/2)^k$. Thus, we have

$$\begin{aligned} D(\tau_w(T) \wedge r) &= D(\tau_w(T)) \wedge D(r \mid \tau_w(T)) \\ &\geqslant \frac{\varepsilon}{2^{k+1}} \cdot \left(\frac{p}{2}\right)^k = \beta. \end{aligned}$$

It is easy to see that the second statement holds, because by Lemma 8 we have $D(y \mid \tau_w(T) \wedge r) \geqslant p/2$ for any literal y that is irrelevant with $\tau_w(T) \wedge r$. ∎

The next lemma asserts that Algorithm LEARN yields with high probability an hypothesis that belongs to $\mathscr{H}$. In what follows $\mathscr{H}$ is regarded as a random variable that is determined by the algorithm in terms of a positive sample S^+ and a negative sample S^-, drawn according to a p-smooth distribution D.

LEMMA 10. *With probability at least $1 - \delta/2$, the $\mathscr{H}$ that Algorithm* LEARN *computes includes an $\varepsilon/2$-approximation.*

Proof. Let $\tilde{T}$ be taken as in Lemma 4. Let g have the XOR-expansion written as $\bigoplus_{1 \leqslant i \leqslant d} g_i$. As Fig. 2 shows, $\mathscr{H}$ consists of the hypothesis of the form $g'(T')$, where g' and T' are taken by the exhaustive search from $\mathscr{G}_k$ and $\hat{\mathscr{U}}^k$, respectively.

It is easy to see that if $\{\tau_w(T) \mid w \in \text{Min}(L_i)\} \subseteq \mathscr{U}_i$ holds for $1 \leqslant i \leqslant d$, then $\hat{\mathscr{U}}^k$ contains $\tilde{T}$ as given in Lemma 4. So since $g(\tilde{T})$ is an $\varepsilon/2$-approximation of $g(T)$ by Lemma 6, it suffices to show that, with probability at least $1 - \delta/2$, the $\mathscr{U}_i$ that Algorithm LEARN computes satisfies

$$\{\tau_w(T) \mid w \in \text{Min}(L_i)\} \subseteq \mathscr{U}_i$$

for any $1 \leqslant i \leqslant d$. Let m, S^+, and S^- be taken as in Algorithm LEARN. Putting $S = (S^+, S^-)$, let $\mathscr{U}_l(S)$ denote the set $\mathscr{U}_l$ that LEARN computes for a pair of samples S. For $1 \leqslant i \leqslant d$, let $C_i(S)$ be the event that $\{\tau_w(T) \mid w \in \text{Min}(L_j)\} \subseteq \mathscr{U}_j(S)$ holds for any $0 \leqslant j \leqslant i$. Note that $C_0(S)$ holds for any sample S. Then, what we will show is written as $\Pr[C_d(S)] \geqslant 1-\delta/2$, or equivalently, $\Pr[\neg C_d(S)] \leqslant \delta/2$. Since V and Y in the algorithm are determined by the sample S, w' in $\mathrm{Min}(L_{i-1})$ and r in $\mathscr{T}_{n,\leqslant k}$, these are written as random variables as

$$V_{w',r}(S) = \{v \in S^{\mathrm{parity}(i)} \mid (\tau_w(T) \wedge r)(v) = 1\}$$

and

$$Y_{w',r}(S) = \{y \in X_n \cup \neg X_n \mid \forall v \in V_{w',r}(S), y(v) = 1\},$$

where parity(i) denotes $+$ when the integer i is odd, and $-$ otherwise. Since W is downward closed, for any w in $\mathrm{Min}(L_i)$ there exist w' and r in R_w that satisfy the equality of Lemma 7. These w' and r are denoted by w_w and r_w, respectively, so we have

$$\tau_w(T) \wedge r_w = \tau_{w_w}(T) \wedge r_w \wedge (g \circ T)^{g(w)}.$$

Note that, when $i=1$, w_w becomes 0^k, and hence, the right-hand side of the above equality turns out to be $r_w \wedge (g \circ T)^{g(w)}$.

It is easy to see that we have

$$\begin{aligned}
&\Pr[C_{i-1}(S) \text{ and } \neg C_i(S)] \\
&\quad \leqslant \Pr\left[\exists w \in \mathrm{Min}(L_i), |V_{w_w,r_w}(S)| < \tfrac{3}{4}\beta m \text{ or } \bigwedge(Y_{w_w,r_w}(S)) \neq \tau_w(T) \wedge r_w\right] \\
&\quad \leqslant \sum_{w \in \mathrm{Min}(L_i)} \Bigg(\Pr[|V_{w_w,r_w}(S)| < \tfrac{3}{4}\beta m] \\
&\qquad + \Pr\left[\bigwedge(Y_{w_w,r_w}(S)) \neq \tau_w(T) \wedge r_w \,\middle|\, |V_{w_w,r_w}(S)| \geqslant \tfrac{3}{4}\beta m\right]\Bigg).
\end{aligned}$$

Since $|\mathrm{Min}(L_i)| \leqslant 2^k$ and $d \leqslant k+1$, it suffices to verify

$$\Pr\left[|V_{w_w,r_w}(S)| < \frac{3}{4}\beta m\right] \leqslant \frac{\delta}{2^{k+2}(k+1)} \tag{1}$$

and

$$\Pr\left[\bigwedge(Y_{w_w,r_w}(S)) \neq \tau_w(T) \wedge r_w \,\middle|\, |V_{w_w,r_w}(S)| \geqslant \frac{3}{4}\beta m\right] \leqslant \frac{\delta}{2^{k+2}(k+1)}. \tag{2}$$

Since $D(\tau_{w_w}(T) \wedge r_w \wedge (g \circ T)^{g(w)}) = D(\tau_w(T) \wedge r_w) \geqslant \varepsilon p^k/2^{k+1} = \beta$ holds by Lemma 7 and Lemma 9, we have in view of Lemma 1,

$$\begin{aligned}
&\Pr\left[|V_{w_w,r_w}(S)| < \frac{3}{4}\beta m\right] \\
&\quad = \Pr\left[|V_{w_w,r_w}(S)| < \left(1-\frac{1}{4}\right)\beta m\right] \leqslant e^{-(1/4)^2 m\beta/2} \leqslant \frac{\delta}{2^{k+2}(k+1)},
\end{aligned}$$

verifying inequality (1).

In view of Lemma 1 it is easy to verify that for any y in $(X_n \cup \neg X_n) \backslash \text{lit}(\tau_w(T) \wedge r_w)$,

$$\Pr[\forall v \in V_{w_w, r_w}(S),\ y(v) = 1 \mid |V_{w_w, r_w}(S)| \geqslant m'] \leqslant (1 - p/2)^{m'} \leqslant e^{-pm'/2}.$$

So, putting $m' = (3/4)\, \beta m \geqslant (3/4)\, \beta (8/3\beta p) \ln(2^{k+3}(k+1)n/\delta)$, we have

$$\begin{aligned}
&\Pr\left[\bigwedge (Y_{w_w, r_w}(S)) \neq \tau_w(T) \wedge r_w \,\middle|\, |V_{w_w, r_w}(S)| < \frac{3}{4} \beta m \right] \\
&\quad = \Pr[\exists y \in (X_n \cup \neg X_n) \backslash \text{lit}(\tau_w(T) \wedge r_w),\ \forall v \in V_{w_w w, r_w}(S),\ y(v) \\
&\quad = 1 \mid |V_{w_w, r_w}(S)| \geqslant m'] \\
&\quad \leqslant 2n \Pr[\forall v \in V_{w_w, r_w}(S),\ y(v) = 1 \mid |V_{w_w, r_w}(S)| \geqslant m'] \\
&\quad \leqslant 2n \cdot \frac{\delta}{2^{k+3}(k+1)n} = \frac{\delta}{2^{k+2}(k+1)}.
\end{aligned}$$

This completes the proof of inequality (2) and, hence, the lemma. ∎

LEMMA 11. *Algorithm LEARN outputs, with probability at least $1 - \delta$, an hypothesis in $\mathscr{F}_{k\text{-term}}$ that ε-approximates the target function.*

Proof. Let $\mathscr{H}$ denote the collection of hypotheses that LEARN computes. An hypothesis h is called to pass the check if both $|\{v \in S^+ \mid h(v) = 0\}| < \frac{3}{4}\varepsilon m$ and $|\{v \in S^- \mid h(v) = 1\}| < \frac{3}{4}\varepsilon m$ hold. Then, in view of Lemma 10 we have

$$\begin{aligned}
&\Pr[\text{LEARN does not output an } \varepsilon\text{-approximation}] \\
&\leqslant \Pr[\mathscr{H} \text{ contains no } \varepsilon/2\text{-approximation}] \\
&\quad + \Pr[\text{“}\mathscr{H} \text{ contains } \varepsilon/2\text{-approximation } h_{\leqslant \varepsilon/2}\text{” and} \\
&\qquad \text{“the hypothesis } h_{\leqslant \varepsilon/2} \text{ does not pass the check or} \\
&\qquad \text{there exists a hypothesis in } \mathscr{H} \text{ with error} \\
&\qquad \text{more than or equal to } \varepsilon \text{ that passes the check”}] \\
&\leqslant \delta/2 + \Pr[\text{An } \varepsilon/2\text{-approximation does not pass the check}] \\
&\quad + (|\mathscr{H}| - 1) \cdot \Pr[\text{A hypothesis with error more than} \\
&\qquad \text{or equal to } \varepsilon \text{ passes the check}].
\end{aligned}$$

Since $|\mathscr{U}_i| \leqslant |\mathscr{U}_{i-1}| \cdot |\{\rho \subseteq \text{lit}(r) \mid r \in \mathscr{T}_{n, \leqslant k}\}| \leqslant (2n)^k\, |\mathscr{U}_{i-1}|$ for $1 \leqslant i \leqslant k+1$, we have $|\mathscr{U}| = \sum_{1 \leqslant i \leqslant k+1} |\mathscr{U}_l| \leqslant 2(2n)^{k(k+1)}$. We therefore have $|\mathscr{H}| = |\mathscr{G}_k| \cdot |\hat{\mathscr{U}}^k| \leqslant 2^{2^k} \cdot (|\mathscr{U}|^{2^{k-1}})^k \leqslant (2n)^{2^{k+2}k^3}$. Hence, we have

$$m \geqslant \frac{24}{\varepsilon} \ln \frac{(2n)^{2^{k+4}k^3}}{\delta} \geqslant \frac{24}{\varepsilon} \ln \frac{|\mathscr{H}|}{\delta}.$$

So, since $3\varepsilon m/4 = (1+1/2)\cdot m\cdot(\varepsilon/2)$ and $m \geqslant (24/\varepsilon)\ln(|\mathscr{H}|/\delta)$, it is easy to see, using Lemma 1, that

$$\begin{aligned}&\Pr[\text{An } \varepsilon/2\text{-approximation does not pass the check}]\\ &\leqslant GE\left(\frac{\varepsilon}{2}, m, \frac{1}{2}\right)\\ &\leqslant e^{-(1/2)^2\cdot(\varepsilon m/2)\cdot(1/3)} = e^{-\varepsilon m/24}\\ &\leqslant \frac{\delta}{2\,|\mathscr{H}|}.\end{aligned}$$

On the other hand, since $3\varepsilon m/4 = (1-1/4)\,\varepsilon m$, we have, in view of Lemma 1, that

$$\begin{aligned}&\Pr[\text{A hypothesis with error more than or equal to } \varepsilon \text{ passes the check}]\\ &\leqslant LE\left(\varepsilon, m, \frac{1}{4}\right)\\ &\leqslant e^{-(1/4)^2\cdot\varepsilon m\cdot(1/2)} = e^{-\varepsilon m/32}\\ &\leqslant \frac{\delta}{2\,|\mathscr{H}|}.\end{aligned}$$

Thus, we have

$$\begin{aligned}&\Pr[\text{LEARN does not output as } \varepsilon\text{-approximation}]\\ &\leqslant \delta/2 + \frac{\delta}{2\,|\mathscr{H}|} + (|\mathscr{H}|-1)\frac{\delta}{2\,|\mathscr{H}|} = \delta,\end{aligned}$$

completing the proof. ▮

LEMMA 12. *For fixed k and* $1/p$ *polynomial in n, algorithm* LEARN *halts in time polynomial in n,* $1/\varepsilon$, *and* $1/\delta$.

Sketch of proof. It can be verified that the algorithm halts in time $O((n^{2^k k^3+1}/\varepsilon p^{k+1})\ln(n/\delta))$. ▮

ACKNOWLEDGMENTS

We thank the anonymous referees for many helpful comments on the presentation of the paper.

Received November 20, 1995; final manuscript received November 30, 1998

REFERENCES

Blum, A., and Singh, M. (1990), Learning functions of k terms, *in* "Proceedings, 3rd Workshop on Computational Learning Theory," pp. 144–153, Morgan Kaufmann, San Mateo, CA.

Bshouty, N. H. (1993), Exact learning via the monotone theory, *in* "Proceedings, 35th Symposium on Foundation of Computer Science," pp. 302–311.

Flammini, M., Marchetti-Spaccamela, A., and Kučera, L. (1992), Learning DNF formulae under classes of probability distributions, *in* "Proceedings, 5th Workshop on Computational Learning Theory," pp. 85–92, Morgan Kaufmann, San Mateo, CA.

Kearns, M., Li, M., Pitt, L., and Valiant, L. G. (1987), On the learnability of Boolean formulae, *in* "Proceedings, 19th ACM Symposium on Theory of Computing," pp. 285–295.

Kucera, L., Marchetti-Spaccamela, A., and Protasi, M. (1994), On learning monotone DNF formulae under uniform distributions, *Inform. and Comput.* **110**, 84–95.

Pitt, L., and Valiant, L. G. (1988), Computational limitation on learning from examples, *J. Assoc. Comput. Mach.* **35**(4), 965–984.

Valiant, L. G. (1984), A theory of the learnable, *Comm. ACM* **27**(11), 1134–1142.