数理解析研究所講究録
906 巻 1995 年 236-243

Proper learning algorithm for functions of k terms under smooth distributions

Yoshifumi Sakai Eiji Takimoto Akira Maruoka
Graduate School of Information Sciences, Tohoku University, Sendai 980-77, Japan
Email: { yoshif, t2, maruoka }@ecei.tohoku.ac.jp

Summary: In this paper, we deal with a class written as $\mathcal{F}_1 \circ \mathcal{F}_2{}^k = \{g(f_1(v), \ldots, f_k(v)) \mid g \in \mathcal{F}_1,\ f_1, \ldots, f_k \in \mathcal{F}_2\}$ for classes $\mathcal{F}_1$ and $\mathcal{F}_2$ characterized by "simple" descriptions and study the learnability of $\mathcal{F}_1 \circ \mathcal{F}_2{}^k$ from examples, where $\mathcal{F}_1$ and $\mathcal{F}_2$ are the classes of functions from Σ^k to Σ and those from Σ^n to Σ, where $\Sigma = \{0, 1\}$. Even if both of $\mathcal{F}_1$ and $\mathcal{F}_2$ are learnable, it is hard to learn $\mathcal{F}_1 \circ \mathcal{F}_2{}^k$ in general. For example, in the distribution free setting, it is known to be NP-hard to learn properly k-term DNF, which is represented as $\{\mathrm{OR}\} \circ \mathcal{T}_n{}^k$, where $\mathcal{T}_n$ is the class of all monomials of n variables. In this paper, we first introduce a probabilistic distribution, called a smooth distribution, which is a generalization of q-bounded distribution and product distribution, and define the learnability under this distribution. Then, we give an algorithm that properly learns $\mathcal{F}_k \circ \mathcal{T}_n{}^k$ under smooth distribution in polynomial time for constant k, where $\mathcal{F}_k$ is the class of all Boolean functions of k variables. The class $\mathcal{F}_k \circ \mathcal{T}_n{}^k$ is called the functions of k terms and although it was shown by Blum and Singh to be learned using DNF as a hypothesis class, it remains open whether it is properly learnable under distribution free setting.

1 Introduction

Since Valiant introduced PAC learning model [4], much effort has been devoted to characterize learnable classes of concepts on 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 given by describing Boolean functions such as threshold functions. In each cases, the class is somehow defined by a "simple" description. In this paper, we deal with a class written as $\mathcal{F}_1 \circ \mathcal{F}_2{}^k = \{g(f_1(v), \ldots, f_k(v)) \mid g \in \mathcal{F}_1,\ f_1, \ldots, f_k \in \mathcal{F}_2\}$ for classes $\mathcal{F}_1$ and $\mathcal{F}_2$ characterized by "simple" descriptions and study the learnability of $\mathcal{F}_1 \circ \mathcal{F}_2{}^k$ from examples, where $\mathcal{F}_1$ and $\mathcal{F}_2$ are the classes of functions from Σ^k to Σ and those from Σ^n to Σ, where $\Sigma = \{0, 1\}$. When the target function to be learned is $g(f_1(v), \ldots, f_k(v))$ in $\mathcal{F}_1 \circ \mathcal{F}_2{}^k$ and both of g and $f_1, \ldots, f_k$ are unknown, in general it is impossible to determine the values of $f_1(v), \ldots, f_k(v)$ even if pairs $(v, g(f_1(v), \ldots, f_k(v)))$ are given as examples for sufficiently many v's in Σ^n. Hence, even if both of $\mathcal{F}_1$ and $\mathcal{F}_2$ are learnable, it is hard to learn $\mathcal{F}_1 \circ \mathcal{F}_2{}^k$ in general. For example, in the distribution free setting, it is NP-hard to learn properly k-term DNF, which is represented as $\{\mathrm{OR}\} \circ \mathcal{T}_n{}^k$, where $\mathcal{T}_n$ is the class of all monomials of n variables [2, 3].

Blum and Singh [1] studied the learnability of the class $\mathcal{F}_k \circ \mathcal{T}_n{}^k$, denoted $\mathcal{F}_{k\_\mathrm{term}}$, where $\mathcal{F}_k$ is the class of all Boolean functions of k variables, and showed that, for constant k, $\mathcal{F}_{k\_\mathrm{term}}$ 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, proper learning $\{g\} \circ \mathcal{T}_n{}^k$ is NP-hard.

In this paper, we first introduce a probabilistic distribution, called a smooth distribution, which is a generalization of q-bounded distribution and product distribution, and define the learnability under this distribution. Then, we give an algorithm that properly learns $\mathcal{F}_{k\_\mathrm{term}}$ under smooth distribution in polynomial time for constant k.

2 Preliminaries

In this extended abstract we follow the standard terminologies in PAC learning model unless otherwise stated. Obtaining positive and negative examples of a target function f through oracles POS() and NEG(), a learning algorithm is expected to produce a hypothesis h that approximates the target function f. A target function f and a hypothesis h are assumed to be Boolean functions of variables $x_1, \ldots, x_n$.

In the following, we often identify a Boolean formula with the Boolean function that it represents. So we regard the class of Boolean formulas as the corresponding class of Boolean functions. For a given Boolean formula (or the corresponding Boolean function) f, let $\mathcal{D}_f$ denote the set of all pairs (D^+, D^-) of probability distribution D^+ on the set of all positive examples of f and probability distribution D^- on the set of all negative examples of f. For a class $\mathcal{F}$ of Boolean formulas (or the corresponding class of Boolean functions), let $\mathcal{D}_{\mathcal{F}}$ denote $\bigcup_{f \in \mathcal{F}} \mathcal{D}_f$. Oracles generate examples independently according to some probability distributions D^+ and D^- for some (D^+, D^-) in $\mathcal{D}_f$. In PAC learning model, the examples are usually assumed to be generated according to either an arbitrary distribution or a uniform distribution. In this paper we assume more general setting where the class of distributions according to which examples are drawn is taken arbitrarily as in Definition 2 below. Let $\Sigma = \{0, 1\}$ and let D be a distribution on subset V of Σ^n. For a vector v in Σ^n and a subset $V' \subseteq \Sigma^n$, let $D(v)$ denote the probability assigned to v under D and $D(V')$ denote $\sum_{v \in V' \cap V} D(v)$. A Boolean function (formula) g also represents the set of vectors v in Σ^n such that $g(v) = 1$. So $D(g)$ represents $\sum_{f(g)=1} D(v)$ and $g \subseteq g'$ means $\{v \mid g(v) = 1\} \subseteq \{v \mid g'(v) = 1\}$. For Boolean functions g and g', $D(g \mid g')$ denotes $D(g \wedge g')/D(g')$. The size of a Boolean function g is the number of symbols appearing in the shortest description of g under some reasonable encoding. Given a class of Boolean functions $\mathcal{F}$, $\mathcal{F}_{n,s}$ denotes the set of Boolean functions of n variables with size at most s in $\mathcal{F}$.

Definition 1 *Let f be a Boolean function, and let $(D^+, D^-) \in \mathcal{D}_f$. A Boolean function h ε-approximates f under (D^+, D^-) if $D^+(f - h) < \varepsilon$ and $D^-(h - f) < \varepsilon$ hold.*

Definition 2 *Let $\mathcal{F}$ be a class of Boolean functions, and let $\mathcal{D}$ be a subset of $\mathcal{D}_{\mathcal{F}}$. An algorithm L learns $\mathcal{F}$ under $\mathcal{D}$ if and only if for any positive integers n, s, any target function f in $\mathcal{F}_{n,s}$, any real numbers ε, δ with $0 < \varepsilon$, $\delta < 1$, and any pair of probability distributions (D^+, D^-) in $\mathcal{D} \cap \mathcal{D}_f$, 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 steps at most some polynomial in n, s, $1/\varepsilon$ and $1/\delta$, and outputs a hypothesis h in $\mathcal{F}_n$ that, with probability at least $1 - \delta$, ε-approximates f under (D^+, D^-). Furthermore, if there exists a learning algorithm for F under $\mathcal{D}$, then F is called learnable under $\mathcal{D}$.*

For a vector v in Σ^n and an integer $1 \leq i \leq n$, 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 $(0, 0, \ldots, 0)$ and $(1, 1, \ldots, 1)$ in Σ^n, respectively. For v and v' in Σ^n, let $v \leq v'$ denote the condition that $v_i \leq v'_i$ for any $1 \leq i \leq n$, and let $v < v'$ denote the condition that $v \leq v'$ and $v \neq v'$. For any subset V of Σ^n, let $Min_{\leq} V$ denote a subset of V defined as

$$Min_{\leq} V = \{v \in V \mid \forall v' \in V - \{v\} \quad v' \not\leq v\},$$

and let $Mon(V)$ denote a monotone Boolean function of n variables defined as

$$Mon(V)(v) = \begin{cases} 1 & \exists v' \in V \quad v' \leq v \\ 0 & \text{otherwise.} \end{cases}$$

Let X_n denote the set of Boolean variables $x_1, \ldots, x_n$. Let Y_n denote a set $X_n \cup \{\neg x_i \mid x_i \in X_n\}$. Let $\mathcal{F}_n$ denote the set of all Boolean functions of n variables. Let TRUE and FALSE denote constant functions that take 1 and 0, respectively. A conjunction of literals is called a term. Let $\mathcal{T}_n$ denote the set of all terms of literals Y_n. For a positive integer k, $\mathcal{T}_{n,\leq k}$ denote the set of terms t of n variables with

$|lit(t)| \leq k$. For a term t, $lit(t)$ denotes the set of literals that appear in t. For any vector v in Σ^n, σ_v and τ_v denote terms of n variables defined as

$$\sigma_v = \bigwedge_{i \in true(v)} x_i \wedge \bigwedge_{i \in false(v)} \neg x_i,$$

$$\tau_v = \bigwedge_{i \in true(v)} x_i \qquad (e.g., \quad \tau_{0^n} = \text{TRUE}),$$

respectively.

For a Boolean function g of k variables and k-tuple $T = (t_1, \ldots, t_k)$ of terms of n variables, $g(T)$ denotes a Boolean function of n variables that takes value $g(t_1(v), \ldots, t_k(v))$ for a vector v in Σ^n. A Boolean function that can be represented as $g(T)$ for some g in $\mathcal{F}_k$ and for some $T = (t_1, \ldots, t_k)$ in ${\mathcal{T}_n}^k$ is called a function of k terms, and $\mathcal{F}_{k\_\mathrm{term}}$ denotes the class of functions of k terms. For example, the class $\mathcal{F}_{2\_\mathrm{term}}$ includes the function $(x_1 \wedge \neg x_2) \oplus (x_3 \wedge x_4 \wedge x_5)$, where $\oplus$ denotes the exclusive OR function. A function $g(T)$ in $\mathcal{F}_{k\_\mathrm{term}}$ can be represented as the composed function $g \circ T$ of function g from Σ^k to Σ and function T from Σ^n to Σ^k. Similarly, in the following, we use notations such as $\sigma_v(T)$, $\tau_v(T)$, $\sigma_v \circ T$ and $\tau_v \circ T$.

Definition 3 *For positive integer n and real number $0 < p \leq 1$, probability distribution D on Σ^n is p-smooth if, for any vectors v and v' in Σ^n with Hamming distance 1, $D(v)/D(v') \geq p$ holds . For a Boolean function f of n variables and real number $0 < p \leq 1$, a pair of probability distributions (D^+, D^-) in $\mathcal{D}_f$ is p-smooth if there exists a p-smooth probability distribution D on Σ^n such that $D^+(v) = D(v)/D(f)$ for any positive vector v of f, and $D^-(v) = D(v)/D(\neg f)$ for any negative vector v of f. Let $\mathcal{S}_{f,p}$ denote the class of all p-smooth pairs (D^+, D^-) of $\mathcal{D}_f$. Furthermore, for a class $\mathcal{F}$ of Boolean functions, let $\mathcal{S}_{\mathcal{F},p}$ denote the class $\bigcup_{f \in \mathcal{F}} \mathcal{S}_{f,p}$, and $\mathcal{S}_{\mathcal{F},p}$ is simply written as $\mathcal{S}_p$ when no confusion arises.*

3 Learning algorithm

A learning algorithm is assumed to get information about a target function $g \circ T$ through positive and negative examples of $g \circ T$. But, in general, it is impossible to know the value of $T(v)$ by observing the examples of $g \circ T$. To overcome the difficulty, the learning algorithm presented in this paper finds an ε-approximation of $g \circ T$ as follows. Instead of trying to find T, the algorithm seeks for a k-tuple of terms, denoted $\tilde{T}_{W,g,T}$, which can be found by observing sufficiently many examples of $g \circ T$. The k-tuple $\tilde{T}_{W,g,T}$ is determined by $W \subseteq \Sigma^k$, $g \in \mathcal{F}_k$, and $T = (t_1, \ldots, t_k) \in {\mathcal{T}_n}^k$. As Lemma 2 states, it turns out that there exists a function, denoted $\tilde{g}_{W,g}$, in $\mathcal{F}_k$ such that $\tilde{g}_{W,g} \circ \tilde{T}_{W,g,T}$ ε-approximates $g \circ T$. The fact that function $\tilde{g}_{W,g}$, which takes the same value as g on W (Proposition 1), is represented as the exclusive OR of at most $(k+1)$ monotone Boolean functions, guarantees that the learning algorithm can find $\tilde{T}_{W,g,T}$ in feasible time. Actually, the learning algorithm finds $\tilde{g}_{W,g} \circ \tilde{T}_{W,g,T}$ that $\varepsilon/2$-approximates $g \circ T$. In the following, since g, T and smooth distribution (D^+, D^-) are assumed to be fixed arbitrarily, we may drop suffices such as g, T and (D^+, D^-), e.g., $\tilde{g}_{W,g}$ and $\tilde{T}_{W,g,T}$ are simply written as $\tilde{g}_W$ and $\tilde{T}_W$, respectively. The learning algorithm first finds a set $\hat{\mathcal{U}}^k$ of k-tuples of terms that includes $\tilde{T}_W$ for appropriate W such that $\tilde{g}_W \circ \tilde{T}_W$ $\varepsilon/2$-approximates $g \circ T$, and then finds g' in $\mathcal{F}_k$ and U in $\hat{\mathcal{U}}^k$ by exhaustive search such that $g' \circ U$ approximates $g \circ T$ with sufficient accuracy.

In this section, we first define $\tilde{g}_W$ and $\tilde{T}_W$ mentioned above, and then explain how the algorithm finds these functions.

A Boolean function g in $\mathcal{F}_k$, k-tuple $T = (t_1, \ldots, t_k)$ in ${\mathcal{T}_n}^k$ and p-smooth distribution (D^+, D^-) in $\mathcal{D}_{g \circ T}$ are assumed to be fixed arbitrarily. Let W be any subset of Σ^k. Let subsets $M_{W,0}, M_{W,1}, \ldots, M_{W,k+1}$ of Σ^k be defined as

$$M_{W,0} = \{0^k\},$$

and for $1 \leq l \leq k+1$,

$$M_{W,l} = Min_{\leq} \left\{ w' \in W \;\middle|\; \begin{array}{l} \exists w \in M_{W,l-1} \\ w < w',\ g(w) \neq g(w') \end{array} \right\}.$$

Furthermore, let $d_{W,l}$ be defined to be $Mon(M_{W,l})$ for $0 \leq l \leq k+1$. It is clear that there exists $1 \leq l' \leq k+1$ such that TRUE $= d_{W,0} \supsetneq d_{W,1} \supsetneq \cdots \supsetneq d_{W,l'} = d_{W,l'+1} = \cdots = d_{W,k+1} =$ FALSE, and hence, W is partitioned into the blocks

$$\{W \cap (d_{W,0} - d_{W,1}), W \cap (d_{W,1} - d_{W,2}), \ldots, W \cap (d_{W,l'-1} - d_{W,l'})\}.$$

Furthermore, by definitions, it is easy to see that g takes the same value on each block and the opposite values on any neighboring blocks. Let $\tilde{g}_W$ denote the Boolean function of k variables defined as

$$\tilde{g}_W = g(0^k) \oplus \bigoplus_{1 \leq l \leq k} d_{W,l}.$$

Then since, for any $0 \leq j \leq l'-1$ and any vector w in $W \cap (d_{W,j} - d_{W,j+1})$,

$$\tilde{g}_W(w) = g(0^k) \oplus \bigoplus_{1 \leq l \leq j} d_{W,l}(w) = g(0^k) \oplus \overbrace{1 \oplus \cdots \oplus 1}^{j} = g(w),$$

the following proposition holds.

Proposition 1 *For any vector w in W, $g(w) = \tilde{g}_W(w)$.*

Let $sign_g$ denote the function defined as $sign_g(j) = g(0^k) \oplus \overbrace{1 \oplus \cdots \oplus 1}^{j}$ for $1 \leq j \leq k$. Then $sign_g(j)$ represents the value that g takes on the region $W \cap (d_{W,j} - d_{W,j+1})$.

Let M_W denote $\bigcup_{1 \leq i \leq k} M_{W,i}$. For $1 \leq i \leq k$, $\tilde{t}_{W,i}$ denotes a term defined as

$$\tilde{t}_{W,i} = \bigwedge_{y \in Y} y, \qquad \text{where } Y = \bigcap_{\substack{w \in M_W \\ w_i = 1}} lit(\tau_w(T)).$$

In the above definition, $\tilde{t}_{W,i}$ denotes FALSE when $w_i = 0$ for any vector w in M_W. Let

$$\tilde{T}_W = (\tilde{t}_{W,1}, \ldots, \tilde{t}_{W,k}).$$

Proposition 2 *For any vector w in M_W, $\tau_w(T) = \tau_w(\tilde{T}_W)$.*

Proof: It suffices to show that $lit(\tau_w(T)) = lit(\tau_w(\tilde{T}_W))$. Recalling $T = (t_1, \ldots, t_k)$, we have $\tau_w(T) = \bigwedge_{w_i=1} t_i$. Since $lit(t_i) \subseteq lit(\tau_{w'}(T))$ holds for any $1 \leq i \leq k$ and any w' in Σ^k with $w'_i = 1$, we have $lit(t_i) \subseteq lit(\tilde{t}_{W,i})$, which implies $lit(\tau_w(T)) = \bigcup_{w_i=1} lit(t_i) \subseteq \bigcup_{w_i=1} lit(\tilde{t}_{W,i}) = lit(\tau_w(\tilde{T}_W))$. On the other hand, since $w \in M_W$, we have $lit(\tau_w(T)) \supseteq \bigcap_{w' \in M_W, w'_i=1} lit(\tau_{w'}(T)) = lit(\tilde{t}_{W,i})$ for any i with $w_i = 1$. Therefore, $lit(\tau_w(T)) \supseteq \bigcup_{w_i=1} lit(\tilde{t}_{W,i}) = lit(\tau_w(\tilde{T}_W))$. □

Since g and $\tilde{g}_W$ take the same value on W, $\tilde{g}_W \circ \tilde{T}_W$ ε-approximates $g \circ T$ when W mentioned above includes all vectors w with $D^{g(w)}(\{v \mid T(v) = w\}) \geq \varepsilon/2^k$ (Lemma 2), where D^1 and D^0 denote D^+ and D^-, respectively. In order to show this, we need to define some notations as follows. Let $range(T)$ denote set $\{w \in \Sigma^k \mid \exists v \in \Sigma^n \quad w = T(v)\}$, and let $range^+(T) = range(T) \cup g$ and $range^-(T) = range(T) \cap (\neg g)$. Then $range(T)$ is partitioned into $range^+(T)$ and $range^-(T)$. Let $range_{\geq q}(T)$ denote the subset $\{w \in range(T) \mid D^{g(w)}(\sigma_w(T)) \geq q\}$, where $D^{g(w)}(\sigma_w(T))$ denotes $D^{g(w)}(\{v \in \Sigma^n \mid T(v) = w\})$. Let $range^+_{\geq q}(T) = range_{\geq q}(T) \cap g$ and $range^-_{\geq q}(T) = range_{\geq q}(T) \cap (\neg g)$. Then it is easy to see the following lemma.

Lemma 1 *If a Boolean function h satisfies $(g \circ T)(v) = h(v)$ for any w in $range_{\geq \varepsilon/2^k}(T)$ and any v in Σ^n with $T(v) = w$, then h ε-approximates $g \circ T$ under (D^+, D^-).*

Using Propositions 1, 2, and Lemma 1, we can show the following lemma.

Lemma 2 *If $range_{\geq \varepsilon/2^k}(T) \subseteq W$, then $\tilde{g}_W \circ \tilde{T}_W$ ε-approximates $g \circ T$ under (D^+, D^-).*

Proof: Let w be any vector in W and let j be a suffix such that $w \in d_{W,j} - d_{W,j+1}$, that is, $d_{W,j}(w) = 1$ and $d_{W,j+1}(w) = 0$. Since $w \in W$, we have $g(w) = \tilde{g}_W(w)$ by Proposition 1. Therefore, since $\tilde{g}_W$ takes the same value on $d_{W,j} - d_{W,j+1}$ and $w \in d_{W,j} - d_{W,j+1}$, we have $g(w) = \tilde{g}_W(w')$ for any w' in $d_{W,j} - d_{W,j+1}$.

Therefore, if $T(v) = w$ implies $\tilde{T}_W(v) \in d_{W,j} - d_{W,j+1}$, then $(g \circ T)(v) = (\tilde{g}_W \circ \tilde{T}_W)(v)$ for any v in Σ^n with $T(v) = w$. That is, for any w in W (and hence, for any w in $range_{\geq \varepsilon/2^k}(T)$), $g \circ T$ and $\tilde{g}_W \circ \tilde{T}_W$ take the same value on $\{v \mid T(v) = w\}$. Thus, by Lemma 1, $\tilde{g}_W \circ \tilde{T}_W$ ε-approximates $g \circ T$ under (D^+, D^-). In the following, we show that $T(v) = w$ implies $\tilde{T}_W(v) \in d_{W,j} - d_{W,j+1}$.

Since w in $Mon(M_{W,j})$, there exists w' in $M_{W,j}$ such that $w' \leq w$. From Proposition 2, we have

$$\tau_w(T) \subseteq \tau_{w'}(T) = \tau_{w'}(\tilde{T}_W) \subseteq Mon(M_{W,j}) \circ \tilde{T}_W = d_{W,j} \circ \tilde{T}_W.$$

On the other hand,

$$d_{W,j+1} \circ T = d_{W,j+1} \circ (t_1, \ldots, t_k) \supseteq d_{W,j+1} \circ (\tilde{t}_{W,1}, \ldots, \tilde{t}_{W,k}) = d_{W,j+1} \circ \tilde{T}_W$$

since, for any $1 \leq i \leq k$, $lit(t_i) \subseteq lit(\tilde{t}_{W,i})$, that is, $t_i \supseteq \tilde{t}_{W,i}$. Therefore we have

$$\begin{aligned} T(v) = w \quad &\Rightarrow \quad (\tau_w \circ T)(v) = 1 \text{ and } (d_{W,j+1} \circ T)(v) = 0 \\ &\Rightarrow \quad (d_{W,j} \circ \tilde{T}_W)(v) = 1 \text{ and } (d_{W,j+1} \circ \tilde{T}_W)(v) = 0 \\ &\Rightarrow \quad ((d_{W,j} - d_{W,j+1}) \circ \tilde{T}_W)(v) = 1 \\ &\Rightarrow \quad \tilde{T}_W(v) \in (d_{W,j} - d_{W,j+1}) \end{aligned}$$

□

Let $f = g \circ T$ be a target function and let W be any subset of Σ^k such that $range_{\geq \varepsilon/2^{k+1}} \subseteq W$. Lemma 2 says that, in order to obtain $\tilde{T}_W = (\tilde{t}_{W,1}, \ldots, \tilde{t}_{W,k})$ such that $\tilde{g}_W \circ \tilde{T}_W$ $\varepsilon/2$-approximates f, it is sufficient to find $\tau_w(T)$ for each w in M_W, because $\tilde{t}_{W,i} = \wedge \left(\bigcap_{w \in M_W, w_i = 1} lit(\tau_w(T)) \right)$.

To find $\tau_w(T)$ for each w in M_W, the algorithm finds sets $\{\tau_w(T) \mid w \in M_{W,l}\}$ for $l = 0, 1, \ldots, k$, repeatedly. More precisely, to find $\tau_{w'}(T)$ for each w' in $M_{W,l}$, the algorithm uses $\tau_w(T)$ previously found for w in $M_{W,l-1}$ with $w < w'$. Since $w < w'$ holds,

$$lit(\tau_{w'}(T)) = lit(\tau_w(T)) \cup \bigcup_{\substack{1 \leq i \leq k \\ w_i = 0, w'_i = 1}} lit(t_i).$$

In order to find $\tau_{w'}(T)$, the algorithm tries to find a set V consisting of sufficient number of vectors generated according to $D^{g(w')}$ with $\sigma_{w'}(T)(v) = 1$ (that is, $T(v) = w'$), and to compute $\wedge\{y \in Y_n \mid \forall v \in V \quad y(v) = 1\}$. There is, however, no obvious way to know the value of $T(v)$ for vector v. So we explore conditions such that $T(v) = w'$ holds for some w' satisfying the conditions mentioned above. The conditions have to be expressed in terms of v and $\tau_{w'}(T)$ without referring to $T(v)$. The conditions we notice consist of three conditions. The first condition is $\tau_w(T)(v) = 1$. The second condition is the one that guarantees $t_i(v) = 0$ for all i with $w'_i = 0$. Provided that y_i is chosen from $lit(t_i) - lit(\tau_{w'}(T))$ for each i with $w'_i = 0$, let $r = \bigwedge_{i \in false(w')} \neg y_i$. The second condition we adopt is $r(v) = 1$ for such y_i's which are found by exhaustive search. Then, if v satisfies these two conditions, we can easily see that

$w \leq T(v) \leq w'$ holds. The third condition we take is $f(v) = g(w')$. When w' is the minimal vector among w'' in $range(T)$ such that $g(w'') \neq g(w)$ and that $w'' \geq w$, it follows that $f(v) = g(T(v)) = g(w')$ for $T(v) \geq w$ implies $T(v) \geq w'$. Thus the third condition, together with the first and second conditions, guarantees that $T(v) = w'$ (Lemma 3).

Using these three conditions, the algorithm finds a set V of sufficient number of v's such that $T(v) = w'$ and computes set $\{y \in Y_n \mid \forall v \in V \quad y(v) = 1\}$. Literals in $\{y \in Y_n \mid \forall v \in V \quad y(v) = 1\}$ are candidates for literals corresponding to $\tau_{w'}(T)$, i.e., those appearing in $\bigwedge_{i \in true(w')} t_i$. Since there may be a literal $\neg y_i$ appearing in r but not in $\bigwedge_{i \in true(w')} t_i$, it is necessary to remove all such literals from $\{y \in Y_n \mid \forall v \in V \quad y(v) = 1\}$ to obtain $lit(\tau_{w'}(T))$. In algorithm LEARN given in Figure 1, a possible set of such literals is denoted by ρ.

The argument above suggests to take as W the set, denoted $\hat{W}$, which is defined as follows.

$$\begin{aligned}\hat{W} &= \{w \in range^+(T) \mid \exists w' \in range^+_{\geq \varepsilon/2^{k+1}}(T) \quad w \leq w'\} \\ &\cup \{w \in range^-(T) \mid \exists w' \in range^-_{\geq \varepsilon/2^{k+1}}(T) \quad w \leq w'\}.\end{aligned}$$

Let $child_{\hat{W}}(w)$ denote $Min_{\leq}\{w' \in \hat{W} \mid w' \geq w, g(w') \neq g(w)\}$. Then clearly, for any w' in $M_{\hat{W},l}$, there exists w' in $child_{\hat{W}}(w)$ such that $w \in M_{\hat{W},l-1}$, where $1 \leq l \leq k$. Note that if $w' \in child_{\hat{W}}(w)$, then $\tau_{w'}(T) \subsetneq \tau_w(T)$ holds. Let $\mathcal{R}_w$ be defined as

$$\mathcal{R}_{1^k} = \{\text{TRUE}\},$$

and for w in $\Sigma^k - \{1^k\}$,

$$\mathcal{R}_w = \left\{ r \in \mathcal{T}_{n,\leq k} \;\middle|\; r \neq \text{FALSE}, r = \bigwedge_{i \in false(w)} \neg y_i, y_i \in lit(t_i) - lit(\tau_w(T)) \right\}.$$

Then, we can show the following lemmas.

Lemma 3 *For any vector w in $M_{\hat{W}}$, any vector w' in $child_{\hat{W}}(w)$ and any term r in $\mathcal{R}_{w'}$,*

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

holds, where $(g \circ T)^1$ and $(g \circ T)^0$ denotes $g \circ T$ and $\neg(g \circ T)$, respectively.

Note that the above lemma implies that $D^{g(w')}(\tau_w(T) \wedge r) = D^{g(w')}(\tau_{w'}(T) \wedge r)$, and hence $D^{g(w')}(y \mid \tau_w(T) \wedge r) = 1$ for any y in $lit(\tau_{w'}(T) \wedge r)$.

Lemma 4 *Let $(D^+, D^-) \in \mathcal{S}_{g \circ T, p}$. For any w in $\hat{W}$, any w' in $child_{\hat{W}}(w)$ and r in $\mathcal{R}_{w'}$,*

$$D^{g(w')}(\tau_w(T) \wedge r) \geq \beta$$

holds, and for any x_i with $\{x_i, \neg x_i\} \cap lit(\tau_{w'}(T) \wedge r) = \emptyset$,

$$\gamma \leq D^{g(w')}(x_i \mid \tau_w(T) \wedge r) \leq 1 - \gamma$$

holds, where $\beta = \varepsilon p^k / 2^{2k+1}$ and $\gamma = p/2$.

We are now ready to construct Algorithm LEARN to learn $\mathcal{F}_k \circ \mathcal{T}_n{}^k$ under p-smooth distributions. An outline of the algorithm is given as follows. Algorithm LEARN first obtains samples S^+ of m positive examples and S^- of m negative examples by calling POS() and NEG() m times, respectively, where m is a sufficiently large number. Then, LEARN puts $\mathcal{U}_0 = \{\text{TRUE}\}$, and computes the sets $\mathcal{U}_1, \ldots, \mathcal{U}_k$ such that $\{\tau_w(T) \mid w \in M_{\hat{W},l}\} \subseteq \mathcal{U}_l$ for $1 \leq l \leq k$, repeatedly. For $1 \leq l \leq k$, $\mathcal{U}_l$ is computed by using $\mathcal{U}_{l-1}$ as follows. Assume that LEARN has $\mathcal{U}_{l-1}$ such that $\{\tau_w(T) \mid w \in M_{\hat{W},l-1}\} \subseteq \mathcal{U}_{l-1}$ holds, and

Algorithm LEARN(n,ε,δ): $\quad$ (* $\beta=\varepsilon p^k/2^{2k-1}$, $\gamma=p/2$ *)
begin
$\quad m \leftarrow \max\left\{\frac{32}{\beta},\frac{4}{3\beta\gamma},\frac{24}{\varepsilon}\right\}\ln\frac{(2n)^{2^{k+4}k^3}k}{\delta}$;
$\quad S^+,S^- \leftarrow \emptyset$; $\quad$ (* multiset *)
$\quad$**for** m **times do**
$\quad\quad$**begin**
$\quad\quad\quad v \leftarrow \mathrm{POS}()$;
$\quad\quad\quad S^+ \leftarrow S^+ \cup \{v\}$;
$\quad\quad\quad v \leftarrow \mathrm{NEG}()$;
$\quad\quad\quad S^- \leftarrow S^- \cup \{v\}$
$\quad\quad$**end**;
$\quad \mathcal{U}_0 \leftarrow \{\mathrm{TRUE}\}$;
$\quad \mathcal{U}_1,\ldots,\mathcal{U}_k \leftarrow \emptyset$;
$\quad$**for** $l \leftarrow 1$ **step** 1 **until** k **do**
$\quad\quad$**for each** $(z,s,r) \in \{+,-\}\times\mathcal{U}_{l-1}\times\mathcal{T}_{n,\le k}$ **do**
$\quad\quad\quad$**begin**
$\quad\quad\quad\quad V \leftarrow \{v\in S^z \mid (s\wedge r)(v)=1\}$; $\quad$ (* multiset *)
$\quad\quad\quad\quad$**if** $|V|\ge\frac{3}{4}\beta m$ **then**
$\quad\quad\quad\quad\quad$**begin**
$\quad\quad\quad\quad\quad\quad u \leftarrow \wedge\{y\in Y_n \mid \forall v\in V \quad y(v)=1\}$;
$\quad\quad\quad\quad\quad\quad \mathcal{U}_l \leftarrow \mathcal{U}_l \cup \left\{\wedge\big(lit(u)-\rho\big)\,\middle|\,\rho\subseteq lit(r)\right\}$
$\quad\quad\quad\quad\quad$**end**
$\quad\quad\quad$**end**;
$\quad \mathcal{U} \leftarrow \bigcup_{1\le l\le k}\mathcal{U}_l$;
$\quad \hat{\mathcal{U}} \leftarrow \left\{\wedge\left(\bigcap_{u\in\mathcal{U}'} lit(u)\right)\,\middle|\,\mathcal{U}'\subseteq\mathcal{U},|\mathcal{U}'|\le 2^{k-1}\right\}\cup\{\mathrm{FALSE}\}$;
$\quad \mathcal{H} \leftarrow \{g'(U)\mid g'\in\mathcal{F}_k, U\in\hat{\mathcal{U}}^k\}$;
$\quad$**for each** $h\in\mathcal{H}$ **do**
$\quad\quad$**if** $|\{v\in S^+\mid h(v)=0\}|<\frac{3}{4}\varepsilon m$ and $|\{v\in S^-\mid h(v)=1\}|<\frac{3}{4}\varepsilon m$ **then**
$\quad\quad\quad$**output** h
end.

Figure 1: Algorithm LEARN

let w' be any vector in $M_{\hat{W},l}$. There exists w in $M_{\hat{W},l-1}$ such that $w' \in child_{\hat{W}}(w)$. If the parameter (z, s, r) of **for** sentence is $(sign_g(l), \tau_w(T), r_{w'})$ for $r_{w'} \in \mathcal{R}_{w'}$, then, by Lemma 4, the set V of vectors v in $S^{sign_g(l)}$ with $(\tau_w(T) \wedge r_{w'})(v) = 1$ satisfies, with sufficiently high probability, $|V| \geq \frac{3}{4}\beta m$. Then, LEARN computes the set $\{y \in Y_n \mid \forall v \in V \quad y(v) = 1\}$. Since by Lemma 4, for any literal y not in $lit(\tau_{w'}(T) \wedge r_{w'})$, both of the probabilities of $y(v) = 1$ and $y(v) = 0$ are lower bounded by some constant (given as $\gamma = p/2$) when v is generated according to $D^{f(w')}$, a literal in $lit(\tau_{w'}(T) \wedge r_{w'})$, with high probability, does not appear in $\{y \in Y_n \mid \forall v \in V \quad y(v) = 1\}$ when $|V|$ is sufficiently large, which implies $\{y \in Y_n \mid \forall v \in V \quad y(v) = 1\} \subseteq lit(\tau_{w'}(T) \wedge r_{w'})$ with high probability, and hence $\{y \in Y_n \mid \forall v \in V \quad y(v) = 1\} = lit(\tau_{w'}(T) \wedge r_{w'})$. Putting ρ a possible set of literals in $lit(r_{w'})$ but not in $lit(\tau_{w'}(T))$, LEARN produces $\wedge(\{y \in Y_n \mid \forall v \in V \quad y(v) = 1\} - \rho)$ and adds it to $\mathcal{U}_l$. Therefore, since **for** sentence is executed for all the possible combinations of parameters z, s, r in the sets given in the algorithm, we have that, with high probability, $\{\tau_{w'}(T) \mid w' \in M_{\hat{W},l}\} \subseteq \mathcal{U}_l$ holds. Since we start with $\{\tau_w(T) \mid w \in M_{\hat{W},0}\} = \{\text{TRUE}\} = \mathcal{U}_0$, it follows that $\{\tau_{w'}(T) \mid w' \in M_{\hat{W},l}\} \subseteq \mathcal{U}_l$ holds with high probability for $1 \leq l \leq k$. Let $\mathcal{U} = \bigcup_{1 \leq l \leq k} \mathcal{U}_l$. Then, since $\tilde{t}_{W,i} = \wedge\left(\bigcap_{w \in M_W, w_i = 1} lit(\tau_w(T))\right)$ for $1 \leq i \leq k$, $\tilde{t}_{\hat{W},i}$ is represented as $\wedge\left(\bigcap_{u \in \mathcal{U}'} lit(u)\right)$ for some appropriate set $\mathcal{U}'$ of at most 2^{k-1} terms in $\mathcal{U}$. Let $\hat{\mathcal{U}}$ be the set of all possible terms $\wedge(\bigcap_{u \in \mathcal{U}'} lit(u))$ for such $\mathcal{U}'$'s. Finally, LEARN obtains the desired hypothesis by checking all the combinations g' in $\mathcal{F}_k$ and $(\tilde{t}_1, \ldots, \tilde{t}_k)$ in $\hat{\mathcal{U}}^k$ until $g' \circ (\tilde{t}_1, \ldots, \tilde{t}_k)$ approximates $g \circ T$ with sufficient accuracy.

4 Correctness

The correctness of algorithm is verified by the following lemmas, which immediately implies Theorem 1.

Lemma 5 *With probability at least $1 - \delta/2$, $\mathcal{H}$ that Algorithm* LEARN *computes includes an $\varepsilon/2$-approximation of $g \circ T$ in $\mathcal{F}_{k\_\mathrm{term}}$ under (D^+, D^-) in $\mathcal{S}_p$.*

Lemma 6 *If $\mathcal{H}$ that Algorithm* LEARN *computes includes an $\varepsilon/2$-approximation of $g \circ T$ in $\mathcal{F}_{k\_\mathrm{term}}$ under (D^+, D^-) in $\mathcal{S}_p$, then* LEARN *outputs, with probability at least $1 - \delta/2$, h in $\mathcal{F}_{k\_\mathrm{term}}$ that ε-approximates $g \circ T$ under (D^+, D^-).*

Lemma 7 *Algorithm* LEARN *halts in time $O((n^{2^{k+4}k^3}/\varepsilon p^{k+1})\ln(n/\delta))$.*

Theorem 1 *If k is constant and p is bounded from below by the inverse of some polynomial in n, $\mathcal{F}_{k\_\mathrm{term}}$ is learnable under $\mathcal{S}_p$.*

References

[1] A. Blum, M. Singh, Learning functions of k terms, in Proceedings of the 3rd Annual Workshop on Computational Learning Theory, Morgan Kaufmann, 1990, pp.144–153.

[2] M. Kearns, M. Li, L. Pitt and L. G. Valiant, On the Learnability of Boolean Formulae, In proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, pp.285–295.

[3] L. Pitt and L. G. Valiant, Computational limitation on learning from examples, Journal of the ACM, Vol.35, No.4, 1988, pp.965–984.

[4] L. G. Valiant, A theory of the learnable, Communications of the ACM, 27(11), 1984, pp.1134–1142.