<table>
<thead>
<tr>
<th>Title</th>
<th>Proper learning algorithm for functions of $k$ terms under smooth distributions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Sakai, Yoshifumi; Takimoto, Eiji; Maruoka, Akira</td>
</tr>
<tr>
<td>Citation</td>
<td>数理解析研究所講究録 (1995), 906: 236-243</td>
</tr>
<tr>
<td>Issue Date</td>
<td>1995-04</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://hdl.handle.net/2433/59438">http://hdl.handle.net/2433/59438</a></td>
</tr>
<tr>
<td>Right</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Departmental Bulletin Paper</td>
</tr>
<tr>
<td>Textversion</td>
<td>publisher</td>
</tr>
</tbody>
</table>

Kyoto University
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}@ecee.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 $\Sigma^k$ to $\Sigma$ and those from $\Sigma^n$ to $\Sigma$, 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 $\{\text{OR}\} \circ \mathcal{T}_{n}^k$, 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 $\mathcal{F}_k \circ \mathcal{T}_{n}$ 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 $\mathcal{F}_k \circ \mathcal{T}_{n}$ 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 $\Sigma^k$ to $\Sigma$ and those from $\Sigma^n$ to $\Sigma$, 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 $\Sigma^n$. 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 $\{\text{OR}\} \circ \mathcal{T}_{n}^k$, 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 $\mathcal{F}_k \circ \mathcal{T}_{n}$, denoted $\mathcal{F}_{k,\text{term}}$, where $\mathcal{F}_k$ is the class of all Boolean functions of $k$ variables, and showed that, for constant $k$, $\mathcal{F}_{k,\text{term}}$ 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, proper learning $\{g\} \circ \mathcal{T}_{n}$ 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 $\mathcal{F}_{k,\text{term}}$ 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 \( \text{POS}() \) and \( \text{NEG}() \), 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 \( D_f \) 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 \( D_{\mathcal{F}} \) denote \( \bigcup_{f \in \mathcal{F}} D_f \). Oracles generate examples independently according to some probability distributions \( D^+ \) and \( D^- \) for some \((D^+, D^-)\) in \( D_f \). 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 \( \Sigma^n \). For a vector \( v \) in \( \Sigma^n \) 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 \( \Sigma^n \) such that \( g(v) = 1 \). So \( D(g) \) represents \( \sum_{v(g)=1} D(v) \) and \( g \subseteq g' \) means \( \{ v \mid g(v) = 1 \} \subseteq \{ v \mid g'(v) = 1 \} \). For Boolean functions \( g \) and \( g' \), \( D(g \land g')D(g') \) denotes \( D(g \land g')D(g') \). 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 \((D^+, D^-) \in D_f \). A Boolean function \( h \) \( \varepsilon \)-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 \( D \) be a subset of \( D_{\mathcal{F}} \). An algorithm \( L \) learns \( \mathcal{F} \) under \( D \) if and only if for any positive integers \( n, s \), any target function \( f \) in \( \mathcal{F}_{n,s} \), any real numbers \( \varepsilon, \delta \) with \( 0 < \varepsilon, \delta < 1 \), and any pair of probability distributions \((D^+, D^-) \) in \( D \cap D_f \), when \( L \) is given as input \( n, s, \varepsilon \) and \( \delta \) as well as access to \( \text{POS}() \) and \( \text{NEG}() \) that generate positive and negative examples independently according to \( D^+ \) and \( D^- \), respectively, \( L \) halts in steps at most some polynomial in \( n, s, \varepsilon, \delta \), and outputs a hypothesis \( h \) in \( \mathcal{F}_{n,s} \) that, with probability at least \( 1 - \delta \), \( \varepsilon \)-approximates \( f \) under \((D^+, D^-) \). Furthermore, if there exists a learning algorithm for \( F \) under \( D \), then \( F \) is called learnable under \( D \).

For a vector \( v \) in \( \Sigma^n \) and an integer \( 1 \leq i \leq n \), let \( v_i \) denote the \( i \)-th component of \( v \). For a vector \( v \), let \( \text{true}(v) \) and \( \text{false}(v) \) 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 \( \Sigma^n \), respectively. For \( v \) and \( v' \) in \( \Sigma^n \), 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 \( \Sigma^n \), let \( \text{Min}_V \subseteq V \) denote a subset of \( V \) defined as

\[
\text{Min}_V = \{ v \in V \mid \forall v' \in V - \{ v \} \quad v' \not\leq v \},
\]

and let \( \text{Mon}(V) \) denote a monotone Boolean function of \( n \) variables defined as

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

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 \( \text{TRUE} \) and \( \text{FALSE} \) denote constant functions that take 1 and 0, respectively. A conjunction of literals is called a term. Let \( T_n \) denote the set of all terms of literals \( Y_n \). For a positive integer \( k \), \( T_{n, \leq k} \) denote the set of terms of \( t \) of \( n \) variables with
For a term $t$, $lit(t)$ denotes the set of literals that appear in $t$. For any vector $v$ in $\Sigma^n$, $\sigma_v$ and $\tau_v$ denote terms of $n$ variables defined as

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

$$\tau_v = \bigwedge_{i \in \text{true}(v)} x_i \quad (\text{e.g., } \tau_0 = \text{TRUE}),$$

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 $\Sigma^n$. 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 $T_n^k$ is called a function of $k$ terms, and $\mathcal{F}_{k,\text{term}}$ denotes the class of functions of $k$ terms. For example, the class $\mathcal{F}_{2,\text{term}}$ includes the function $(x_1 \land \neg x_2) \oplus (x_3 \land x_4 \land x_5)$, where $\oplus$ denotes the exclusive OR function. A function $g(T)$ in $\mathcal{F}_{k,\text{term}}$ can be represented as the composed function $g \circ T$ of function $g$ from $\Sigma^k$ to $\Sigma$ and function $T$ from $\Sigma^n$ to $\Sigma^k$. 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 $\Sigma^n$ is $p$-smooth if, for any vectors $v$ and $v'$ in $\Sigma^n$ 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 $D_f$ is $p$-smooth if there exists a $p$-smooth probability distribution $D$ on $\Sigma^n$ 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 $S_{f,p}$ denote the class of all $p$-smooth pairs $(D^+, D^-)$ of $D_f$. Furthermore, for a class $\mathcal{F}$ of Boolean functions, let $S_{\mathcal{F},p}$ denote the class $\bigcup_{f \in \mathcal{F}} S_{f,p}$, and $S_{\mathcal{F},p}$ is simply written as $S_p$ 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 $\epsilon$-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 $T = (t_1, \ldots, t_k) \in T_n^k$. As Lemma 2 states, it turns out that there exists a function, denoted $\tilde{g}_{W,g}$, in $\mathcal{F}_k$ such that $\tilde{g}_{W,g} \circ \tilde{T}_{W,g,T}$ is $\epsilon$-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 $\epsilon/2$-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 $U^k$ of $k$-tuples of terms that includes $\tilde{T}_W$ for appropriate $W$ such that $\tilde{g}_W \circ \tilde{T}_W \epsilon/2$-approximates $g \circ T$, and then finds $g'$ in $\mathcal{F}_k$ and $U$ in $U^k$ 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 $T_n^k$ and $p$-smooth distribution $(D^+, D^-)$ in $D_{g,T}$ are assumed to be fixed arbitrarily. Let $W$ be any subset of $\Sigma^k$. Let subsets $M_{W,0}, M_{W,1}, \ldots, M_{W,k+1}$ of $\Sigma^k$ be defined as

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

.$$
and for $1 \leq l \leq k + 1$,
\[
M_{W,l} = \text{Min}_W \left\{ w' \in W \mid \exists w \in M_{W,l-1}, w < w', g(w) \neq g(w') \right\}.
\]
Furthermore, let $d_{W,l}$ be defined to be $\text{Mon}(M_{W,l})$ for $0 \leq l \leq k + 1$. It is clear that there exists $1 \leq l' \leq k + 1$ such that $\text{TRUE} = d_{W,0} \supseteq d_{W,1} \supseteq \cdots \supseteq d_{W,l'} = d_{W,l'+1} = \cdots = d_{W,k+1} = \text{FALSE}$, 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 $W \cap (d_{W,j} - d_{W,j+1})$,
\[
\tilde{g}_W(w) = g(0^k) \oplus \bigoplus_{1 \leq l \leq j} d_{W,l}(w) = g(0^k) \oplus 1 \oplus \cdots \oplus 1 = g(w),
\]
the following proposition holds.

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

Let $\text{sign}_g$ denote the function defined as $\text{sign}_g(j) = g(0^k) \oplus 1 \oplus \cdots \oplus 1$ for $1 \leq j \leq k$. Then $\text{sign}_g(j)$ 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 $1 \leq i \leq k$, $\bar{i}_{W,i}$ denotes a term defined as
\[
\bar{i}_{W,i} = \bigwedge_{y \in Y} y, \quad \text{where} \quad Y = \bigcap_{w \in M_W} \text{lit}(\tau_w(T)).
\]
In the above definition, $\bar{i}_{W,i}$ denotes FALSE when $w_i = 0$ for any vector $w$ in $M_W$. Let
\[
\tilde{T}_W = (\bar{i}_{W,1}, \ldots, \bar{i}_{W,k}).
\]

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

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

Since $g$ and $\tilde{g}_W$ take the same value on $W$, $\tilde{g}_W \circ \tilde{T}_W$ $\varepsilon$-approximates $g \circ T$ when $W$ mentioned above includes all vectors $w$ with $D^\varepsilon(w)(\{v \mid T(v) = w\}) \geq \varepsilon/2^k$ (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 $\text{range}(T)$ denote set $\{w \in \Sigma^k \mid \exists v \in \Sigma^n \quad w = T(v)\}$, and let $\text{range}^+(T) = \text{range}(T) \cup g$ and $\text{range}^-(T) = \text{range}(T) \cap (-g)$. Then $\text{range}(T)$ is partitioned into $\text{range}^+(T)$ and $\text{range}^-(T)$. Let $\text{range}_{\geq q}(T)$ denote the subset $\{w \in \text{range}(T) \mid D^q(w)(\sigma_w(T)) \geq q\}$, where $D^q(w)(\sigma_w(T))$ denotes $D^q(w)(\{v \in \Sigma^n \mid T(v) = w\})$. Let $\text{range}_{\geq q}(T) = \text{range}_{\geq q}(T) \cap g$ and $\text{range}_{\leq q}(T) = \text{range}_{\leq q}(T) \cap (-g)$. 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_{d/2}(T) and any $v$ in $\Sigma^n$ with $T(v) = w$, then $h$ $\epsilon$-approximates $g \circ T$ under $(D^+, D^-)$.

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

Lemma 2 If range_{d/2}(T) $\subseteq W$, then $\tilde{g}_W \circ \tilde{T}_W$ $\epsilon$-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 $\Sigma^n$ with $T(v) = w$. That is, for any $w$ in $W$ (and hence, for any $w$ in range_{d/2}(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, $\tilde{g}_W \circ \tilde{T}_W$ $\epsilon$-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

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

On the other hand, 

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

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

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

\quad \Box

Let $f = g \circ T$ be a target function and let $W$ be any subset of $\Sigma^k$ such that range_{d/2}(T) $\subseteq W$. Lemma 2 says that, in order to obtain $\tilde{T}_W = (i_{W,1}, \ldots, i_{W,k})$ such that $\tilde{g}_W \circ \tilde{T}_W \epsilon/2$-approximates $f$, it is sufficient to find $\tau_w(T)$ for each $w$ in $M_W$, because $i_{W,i} = \Lambda \left( \bigcap_{w \in M_{W,0}, w \neq 1} lit(\tau_w(T)) \right)$. To find $\tau_w(T)$ for each $w$ in $M_W$, the algorithm finds sets $\{\tau_w(T) \mid w \in M_{W,i}\}$ for $i = 0, 1, \ldots, k$, repeatedly. More precisely, to find $\tau_w(T)$ for each $w'$ in $M_{W,i}$, the algorithm uses $\tau_w(T)$ previously found for $w$ in $M_{W,i-1}$ with $w < w'$. Since $w < w'$ holds, 

$$lit(\tau_{w'}(T)) = lit(\tau_w(T)) \cup \bigcup_{1 \leq i \leq k \atop \text{lit}(t_i) \subseteq \text{lit}(i_{W,i})} lit(t_i).$$

In order to find $\tau_{w'}(T)$, the algorithm tries to find a set $V$ consisting of sufficient number of vectors generated according to $D^{\text{al}(w')} \sigma_{w'}(T)(v) = 1$ (that is, $T(v) = w'$), and to compute $\Lambda \{g \in Y_n \mid \forall v \in V \quad g(v) = 1\}$. 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 $r = \Lambda \{g \in Y_n \mid \tau_w(T) \supseteq y_i\}$. 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 $\{y \in Y_n \mid \forall v \in V \ y(v) = 1\}$. Literals in $\{y \in Y_n \mid \forall v \in V \ y(v) = 1\}$ are candidates for literals corresponding to $\tau_w(T)$, i.e., those appearing in $\bigwedge_{i \in \text{false}(w)} \neg y_i$. Since there may be a literal $\neg y_i$ appearing in $r$ but not in $\bigwedge_{i \in \text{true}(w)} \neg y_i$, it is necessary to remove all such literals from $\{y \in Y_n \mid \forall v \in V \ y(v) = 1\}$ to obtain lit($\tau_w(T)$). In algorithm LEARN given in Figure 1, a possible set of such literals is denoted by $p$.

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

$$W = \{w \in \text{range}^+(T) \mid \exists w' \in \text{range}^+_{\geq 2^l/2^{l+1}}(T) \ w \leq w'\}$$

$$\cup \{w \in \text{range}^-(T) \mid \exists w' \in \text{range}^-_{\geq 2^l/2^{l+1}}(T) \ w \leq w'\}.$$

Let $\text{child}_W(w)$ denote $\text{Min}_{\leq} \{w' \in W \mid w' \geq w, g(w') \neq g(w)\}$. Then clearly, for any $w'$ in $M_{W,l}$, there exists $w'$ in $\text{child}_W(w)$ such that $w \in M_{W,l-1}$, where $1 \leq l \leq k$. Note that if $w' \in \text{child}_W(w)$, then $\tau_w(T) \nsubseteq \tau_w (T)$ holds. Let $R_w$ be defined as 

$$R_w = \{r \in T_n, \leq k \mid r \neq \text{FALSE}, r = \bigwedge_{i \in \text{false}(w)} \neg y_i, y_i \in \text{lit}(t_i) \land \text{lit}(\tau_{w'}(T))\}.$$

Then, we can show the following lemmas.

Lemma 3 For any vector $w$ in $M_{W,l}$, any vector $w'$ in $\text{child}_W(w)$ and any term $r$ in $R_w$,

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

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

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

Lemma 4 Let $(D^+, D^-) \in S_{\geq T,p}$. For any $w$ in $W$, any $w'$ in $\text{child}_W(w)$ and $r$ in $R_w$,

$$D^{\#}(\tau_{w'}(T) \land r) \geq \beta$$

holds, and for any $x_i$ with $\{x_i, \neg x_i\} \cap \text{lit}(\tau_w(T) \land r) = \emptyset$,

$$\gamma \leq D^{\#}(x_i | \tau_w(T) \land r) \leq 1 - \gamma$$

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

We are now ready to construct Algorithm LEARN to learn $\mathcal{F}_k \circ T_n^k$ 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 $U_0 = \{\text{TRUE}\}$, and computes the sets $U_1, \ldots, U_k$ such that $\{\tau_{w}(T) \mid w \in M_{W,l}\} \subseteq U_l$ for $1 \leq l \leq k$, repeatedly. For $1 \leq l \leq k$, $U_l$ is computed by using $U_{l-1}$ as follows. Assume that LEARN has $U_{l-1}$ such that $\{\tau_{w}(T) \mid w \in M_{W,l-1}\} \subseteq U_{l-1}$ holds, and
Algorithm LEARN($n, \varepsilon, \delta$): $(\ast \beta = \varepsilon p^k/2^{2k-1}, \gamma = p/2 \ast)$

\begin{align*}
&\text{begin} \\
&m \leftarrow \max \left\{ \frac{32}{3\beta}, \frac{4}{3\beta \gamma}, \frac{24}{\beta \gamma} \right\} \ln \frac{(2\varepsilon k)}{\delta} \\
&S^+, S^- \leftarrow \emptyset; \quad (\ast \text{multiset} \ast) \\
&\text{for } m \text{ times do} \\
&\quad \text{begin} \\
&\quad \quad v \leftarrow \text{POS}(); \\
&\quad \quad S^+ \leftarrow S^+ \cup \{v\}; \\
&\quad \quad v \leftarrow \text{NEG}(); \\
&\quad \quad S^- \leftarrow S^- \cup \{v\} \\
&\quad \text{end}; \\
&\quad \mathcal{U}_0 \leftarrow \{\text{TRUE}\}; \\
&\quad \mathcal{U}_1, \ldots, \mathcal{U}_k \leftarrow \emptyset; \\
&\quad \text{for } l \leftarrow 1 \text{ step } 1 \text{ until } k \text{ do} \\
&\quad \quad \text{for each } (z, s, r) \in \{+, -\} \times \mathcal{U}_{l-1} \times \mathcal{T}_{n, \leq k} \text{ do} \\
&\quad \quad \quad \text{begin} \\
&\quad \quad \quad \quad V \leftarrow \{v \in S^+ \mid (s \wedge r)(v) = 1\}; \quad (\ast \text{multiset} \ast) \\
&\quad \quad \quad \quad \text{if } |V| \geq \frac{3}{4} \beta m \text{ then} \\
&\quad \quad \quad \quad \quad \text{begin} \\
&\quad \quad \quad \quad \quad \quad u \leftarrow \land \{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 \{\land (\text{lit}(u) - \rho) \mid \rho \subseteq \text{lit}(r)\} \\
&\quad \quad \quad \quad \text{end}; \\
&\quad \quad \text{end}; \\
&\quad \mathcal{U} \leftarrow \bigcup_{1 \leq l \leq k} \mathcal{U}_l; \\
&\quad \hat{\mathcal{U}} \leftarrow \left\{ \land \left( \bigcap_{u \in U'} \text{lit}(u) \right) \mid U' \subseteq U, |U'| \leq 2^{k-1} \right\} \cup \{\text{FALSE}\}; \\
&\quad \mathcal{H} \leftarrow \{g(U) \mid g' \in \mathcal{F}_k, U \in \hat{\mathcal{U}}\}; \\
&\quad \text{for each } h \in \mathcal{H} \text{ do} \\
&\quad \quad \text{if } |\{v \in S^+ \mid h(v) = 0\}| < \frac{3}{4} \varepsilon m \text{ and } |\{v \in S^- \mid h(v) = 1\}| < \frac{3}{4} \varepsilon m \text{ then} \\
&\quad \quad \quad \text{output } h \\
&\quad \text{end}. \\
\end{align*}

Figure 1: Algorithm LEARN
let $w'$ be any vector in $M_{W,i}$. There exists $w$ in $M_{W,i-1}$ such that $w' \in \text{child}_{W}(w)$. If the parameter $(z,s,r)$ of for sentence is $(\text{sign}_{y}(I), \tau_{w}(T), r_{w'}(T))$ for $r_{w'} \in R_{w'}$, then, by Lemma 4, the set $V$ of vectors $v$ in $S^{\text{sign}_{y}(I)}$ with $(\tau_{w}(T) \land r_{w'}(T))(v) = 1$ satisfies, with sufficiently high probability, $|V| \geq \frac{3}{2} \beta m$. Then, LEARN computes the set \( \{y \in Y_n \mid \forall v \in V \; y(v) = 1\} \). Since by Lemma 4, for any literal $y$ not in $\text{lit}(\tau_{w}(T) \land r_{w'}(T))$, 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 $D^{(w')}$, a literal in $\text{lit}(\tau_{w}(T) \land r_{w'}(T))$ with high probability, does not appear in $\{y \in Y_n \mid \forall v \in V \; y(v) = 1\}$ when $|V|$ is sufficiently large, which implies \( \{y \in Y_n \mid \forall v \in V \; y(v) = 1\} \subseteq \text{lit}(\tau_{w}(T) \land r_{w'}(T)) \) with high probability, and hence \( \{y \in Y_n \mid \forall v \in V \; y(v) = 1\} \subseteq \text{lit}(\tau_{w}(T) \land r_{w'}(T)) \). Putting $p$ a possible set of literals in $\text{lit}(\tau_{w}(T))$ but not in $\text{lit}(\tau_{w}(T))$, LEARN produces $\bigwedge \{y \in Y_n \mid \forall v \in V \; y(v) = 1\} - p$ and adds it to $U_t$. 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, \( \{\tau_{w}(T) \mid w' \in M_{W,i}\} \subseteq U_t \) holds. Since we start with \( \{\tau_{w}(T) \mid w \in M_{W,0}\} = \{\text{TRUE}\} = U_0 \), it follows that \( \{\tau_{w}(T) \mid w' \in M_{W,i}\} \subseteq U_t \) holds with high probability for $1 \leq i \leq k$. Let $U = \bigcup_{1 \leq i \leq k} U_t$. Then, since $W_{i,j}$ is represented as $\bigwedge \{\bigcap_{u \in U} \text{lit}(u)\}$ for some appropriate set $U'$ of at most $2^{k-1}$ terms in $U$. Let $U'$ be the set of all possible terms $\bigwedge \{\bigcap_{u \in U} \text{lit}(u)\}$ for such $U'$s. Finally, LEARN obtains the desired hypothesis by checking all the combinations $g'$ in $F_k$ and $(i_1, \ldots, i_k)$ in $U^k$ until $g' \circ (i_1, \ldots, i_k)$ 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 $1 - \delta/2$, $H$ that Algorithm LEARN computes includes an $\varepsilon/2$-approximation of $g \circ T$ in $F_{k,\text{term}}$ under $(D^+, D^-)$ in $S_p$.

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

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

**Theorem 1.** If $k$ is constant and $p$ is bounded from below by the inverse of some polynomial in $n$, $F_{k,\text{term}}$ is learnable under $S_p$.

**References**


