数理解析研究所講究録
第 731 巻 1990 年 84-96

A learning algorithm for monotone k-term DNF

Takeshi OHGURO (大黒 毅)　　Akira MARUOKA (丸岡 章)

Department of Information Engineering,
Faculty of Engineering, Tohoku University

Abstract

Recently, Valiant introduced a computational model of learning, and gave a precice definition of learnability. Since then, much effort has been devoted to characterize learnable classes of concepts on this model. In this paper, we give, based on the uniform distribution model, a polynomial time algorithm that learns k-term MDNF, the class of monotone disjunctive normal formulae with at most k terms. This algorithm uses only positive examples and output hypothesis with one-sided-error. This result should be contrasted with the fact that the same class is not learnable in the distribution free setting. Based on the uniform distribution model, learning algorithms for k-term MDNF were given in [GM,KMP]. But these algorithms use both positive and negative examples. Consequently error of these algorithms is two-sided. Furthermore, the algorithm proposed in this paper is easily modified to learn the same class in the presence of errors in the examples.

1 Introduction

Recently, Valiant introduced a computational model of learning, and gave a precice definition of learnability based on the model [V84]. A class of Boolean formulae is said to be learnable if there exists a polynomial time algorithm to learn any formula in the class: With access to oracles that give some partial information about an unknown target formula in the class, the learning algorithm outputs a Boolean formula that is, with high likelihood, a reasonably accurate approximation to the target. Since Valiant has proposed this learnability model, much effort has been devoted to characterize learnable classes.

In particular, several algorithms to learn classes of Boolean formulae from examples have been studied (see [KLPV87a,H] for example). Among these, the problem of learning the class of disjunctive normal forms (DNF) seems to be important. But whether DNF is learnable from examples is open in the distribution free model. The same problem remains open even if we restrict ourselves to the distribution specific model, where positive or negative examples of f are generated according to the uniform probability distributions.

In this paper we give, based on the uniform distribution model, a polynomial time algorithm to learn k-term MDNF, the class of monotone disjunctive normal formulae with at most k terms, from positive examples. This result should be contrasted with the fact that the same class is not learnable in the distribution free setting [PV].

Based on the uniform distribution model, learning algorithms for k-term MDNF have been proposed in literature. In [GM] a learning algorithm for k-term MDNF is given. A more natural algorithm was proposed in [KMP]. But the crucial part of the algorithm, hence the probability analysis to show its correctness, was not given in the paper.

The idea of the learning algorithm, given in this paper, for k-term MDNF is based on one of the learning algorithm given in [KMP], although one of the key ideas of restricting the domain where a target function is identified was not mentioned explicitly in [KMP]. Using the idea of restrictions carefully, we can construct the learning algorithm that is assumed to use only positive examples, whereas the algorithms given previously in literature were assumed to use both positive and negative examples. Consequently, the type of error for the algorithm of this paper is one-sided, while that for the previous algorithms is two-sided. We give the whole algorithm explicitly together with the proof of its correctness.

Furthermore, the algorithm proposed in this paper is easily modified to learn the same class in the presence of errors in the examples.

2 Preliminaries

We first describe the Valiant's model for learning that we shall use briefly. For more complete discussion and justification of the model, see [PV], [V84] and [KLPV87b].

Let F_n be a set of formulae with variables $\{x_1, \ldots, x_n\}$, and let $F = \bigcup F_n$. Let f be a formula in F_n. For convenience, we regard f in F_n as the corresponding Boolean function with domain $\{0,1\}^n$, and sometimes as the set of vectors such that $\{v | f(v) = 1\}$. A vector $v \in \{0,1\}^n$ is called a positive example (resp. negative example) of f iff $f(v) = 1$ (resp. $f(v) = 0$). D_f^+ (resp. D_f^-) is a probability distribution uniform over all positive (resp. negative) examples of f. D_f^+ (D_f^-) is simply written as D^+ (D^-) when no confusion arises.

In this paper a learning algorithm is assumed to call an oracle POS(f), which produces positive examples of f independently according to the probability distribution D_f^+. POS(f) is simply written as POS when no confusion arises. In more general setting, the learning algorithm is also allowed to call another type of oracles (e.g. NEG(f), which produces negative examples). We adopt the distribution specific model where the probability that examples are generated is taken to be uniform, whereas in the distribution free model the probability is taken to be arbitrary. But in fact, the condition that D^- is uniform is not necessary for the algorithm given in this paper: It works as well when D^- is taken arbitrary.

Definition. A class of formulae F is called *learnable* if there exists an algorithm L_F, with access to POS(f), that satisfies the following conditions:

$\forall n$, $\forall f \in F_n$, $\forall \delta$, $\forall \varepsilon$,

i) L_F runs in time polynomial in n, $\frac{1}{\varepsilon}$ and $\frac{1}{\delta}$.

ii) L_F outputs a formula $g$ in F_n with probability at least $1-\delta$, that satisfies $\sum_{g(v)=0} D^+(v) < \varepsilon$ and $\sum_{g(v)=1} D^-(v) = 0$.

$f$ is called a *target formula* and $g$ is called a *hypothesis*. ε is called a *accuracy parameter* and $\delta$ is called a *confidence parameter*. According to the definition we simply say $F$ is learnable instead of saying $F$ is polinomial time learnable from positive examples under uniform distributions with one-sided-error.

A conjunction of litelals is called a *monomial* (or a *term*). For constant $k$, let $k$-term MDNF denote the class of monotone disjunctive normal forms with up to $k$ terms. Let $Var(t)$ denote the set of variables that appear in monotone term $t$. Let $Term(f)$ denote the set of terms of a formula $f$. Let v_i denotes the ith component of $v$. Likewise, let v_A denote the $i_1, i_2, \ldots$, and i_j th components of $v$, where $A$ is a subset $\{x_{i_1}, \ldots, x_{i_j}\}$ of $\{x_1, \ldots, x_n\}$. $v_A = 0$ means that $v_{i_1} = \cdots = v_{i_j} = 0$.

Next we describe some results useful in probabilistic analysis in this and subsequent sections. Let $LE(p, m, r)$ denote the probability of at most $r$ successes in $m$ independent trials with probability of success at least $p$, and $GE(p, m, r)$ denote the probability of at least $r$ successes with probability of success at most $p$.

Let $b$ be such that $0 \leq b \leq 1$ in the following facts.

Fact 2.1 [KLPV87b] $LE(p, m, (1-b)pm) \leq e^{-b^2 mp/2}$.

Fact 2.2 [KLPV87b] $GE(p, m, (1+b)pm) \leq e^{-b^2 mp/3}$.

Proposition 2.3 [V84] $LE(p, m, r) \leq \delta$ for $m \geq \frac{2}{p}(r + \ln \frac{1}{\delta})$.

Procedure EXAMPLES given in Figure 1 produces a sequence of $m$ positive examples which satisfy the condition $C(v)$. The condition 'TRUE' always holds: It means that there is no spacific condition. $|S|$ denotes the length of the sequence $S$. As in the definition of learnability, $\delta$ is called *confidence parameter* of EXAMPLES.

Proposition 2.4 Let $0 < \delta < 1$. If $\Pr[C(v)] \geq p_c$, then EXAMPLES produces a sequence $S$ of $m$ positive examples which satisfy the condition $C(v)$, with probability at least $1-\delta$. If the condition $C(v)$ is TRUE then the probability is 1.

Proof. Immediate from Proposition 2.3. □

Note that all positive examples in $S$ are independently taken from uniform distribution over the positive examples satisfying the condition $C(v)$.

Procedure FREQ given in Figure 2 can be used to estimate the probability of event $E(v)$ from the fraction of its occurrence in a sequence of trials given by $S$.

```
procedure EXAMPLES($m$, $C(v)$, $p_c$, $\delta$)
begin
    let $S$ be a empty sequence ;
    if $C(v) = \mathrm{TRUE}$ then $m' := m$
    else $m' := \frac{2}{p_c}(m + \ln\frac{1}{\delta})$ ;
    for $c := 1$ to $m'$ do
        begin
            $v := \mathrm{POS}$ ;
            if $C(v)$ holds then add $v$ to $S$ ;
        end
    return $S$ ;
end
```

Figure 1: Procedure EXAMPLES

```
procedure FREQ($S$, $p-\alpha$, $p$, $E(v)$)
begin
    $c := 0$ ;
    foreach $v$ in $S$ do if $E(v)$ holds then $c := c + 1$ ;
    if $c \geq |S|(p - \alpha/2)$ then return "high"
    else return "low" ;
end
```

Figure 2: Procedure FREQ

Lemma 2.5 Let α, p be such that $0 < p \leq 1$, $0 < \alpha \leq \frac{2}{3}p$ and $0 < \delta < 1$. Let S be a sequence of m vectors which are drawn independently according to some distribution (probability is measured by this distribution). Let m satisfies the following condition (2-1).

$$m \geq \max\left\{\frac{8p}{\alpha^2}\ln\frac{1}{\delta},\ \frac{12(p-\alpha)}{\alpha^2}\ln\frac{1}{\delta}\right\} \tag{2-1}$$

Then, if $\Pr[E(v)] \leq p - \alpha$ then FREQ returns "high" with probability at most δ, and if $\Pr[E(v)] \geq p$ then FREQ returns "low" with probability at most δ.

Proof. Assume that $\Pr[E(v)] \leq p - \alpha$. Taking $m = \frac{12(p-\alpha)}{\alpha^2}\ln\frac{1}{\delta}$ and using Fact 2.2 with $b = \alpha/2(p-\alpha)$, it follows that $GE(p-\alpha, m, (p-\alpha/2)m) \leq \exp\left[-\{\frac{\alpha}{2(p-\alpha)}\}^2(p-\alpha)\frac{12(p-\alpha)}{\alpha^2} \cdot \ln\frac{1}{\delta}/3\right] = \delta$. Therefore, the probability that $E(v)$ holds at least $(p-\alpha/2)m$ times among m independent trials is at most δ. Thus FREQ returns "high" with probability at most δ.

For the second part of the Lemma, assume that $\Pr[E(v)] \geq p$. Taking $m = \frac{8p}{\alpha^2}\ln\frac{1}{\delta}$ and using Fact 2.1 with $b = \alpha/2p$ as above, it can be seen that the probability that $E(v)$ holds at most $(p-\alpha/2)m$ times among m independent trials is at most δ. Therefore FREQ returns "low" with probability at most δ.

Since the upper bounds given in Fact 2.1 and Fact 2.2 are monotone decreacing functinos on m, the lemma follows. □

δ is called *confidence parameter* of FREQ. In later sections, Lemma 2.5 will be used to estimate the probability of $E(v)$ from the value procedure FREQ returns: With high confidence the fact that FREQ returns "low" implies that $\Pr[E(v)] < p$, and the fact that FREQ returns "high" implies that $\Pr[E(v)] > p - \alpha$. This is because both of the probability of occurrence of "low" and $\Pr[E(v)] \geq p$, and that of occurrence of "high" and $\Pr[E(v)] \leq p - \alpha$ are at most δ, which will be taken sufficiently small in the following argument. To simplify the argument in section 4, we say that "low" implies $\Pr[E(v)] < p$ and that "high" implies $\Pr[E(v)] > p - \alpha$. We only take care of the probability δ at the end of the argument.

In some cases, we need to estimate the conditional probability of event $E(v)$ under some condition $C(v)$. This is done by combining procedure EXAMPLES and FREQ. In this case, EXAMPLES does not always succeed in producing the desired sequence S, but succeed in with high probability. In section 4, we omit the phrase "with high probability" and treat as if EXAMPLES always produces the desired sequence for simplicity. We only take care of the probability of failure at the end of the argument, as is the case for FREQ.

3 Learning algorithm for k-term MDNF

Before proceeding to describe the learning algorithm L for k-term MDNF in detail, we give an outline of the algorithm together with an idea behind it.

Let $f = t_1 \vee t_2 \vee \cdots \vee t_k$ be a target in k-term MDNF. Without loss of generality, assume that none of the terms of f is redundant in the sense of being implied by the sum of the others. Fact 3.1 below is the basis of the algorithm: In order to find out a term t which is not found yet, the algorithm attempts to determine such a set A as in the fact.

Fact 3.1 Let f be a non-redundunt formula in k-term MDNF. For any term t in $Term(f)$, there exists a set A of variables such that $A \cap Var(t) = \emptyset$, $A \cap Var(t') \neq \emptyset$ for any term t' in $Term(f) - \{t\}$, and $|A| \leq k - 1$.

Suppose that algorithm L succeeded in producing the $k - i$ (> 0) terms in $Term(f)$, $t_1, \ldots, t_{k-i}$, and that it is about to find one of the remaining terms. Let the disjunction of these $k - i$ terms be denoted g. At this point in order to find a term in $Term(f) - Term(g)$, first L calls the procedure SUPPRESS_G. The procedure tries to find $x_{j_1}, \ldots, x_{j_{k-i}}$ belonging to $Var(t_1), \ldots, Var(t_{k-i})$, respectively, so that the region consisting of positive examples v with $v_{\{x_{j_1},\ldots,x_{j_{k-i}}\}} = 0$ is not too small.

When $x_{j_1} \in Var(t_1)$, $\ldots$, $x_{j_{k-i}} \in Var(t_{k-i})$, we say that the valiables $x_{j_1}, \ldots, x_{j_{k-i}}$ suppress the terms $t_1, \ldots, t_{k-i}$. This is because putting $x_{j_1} = 0, \ldots, x_{j_{k-i}} = 0$ makes all of the terms $t_1, \ldots, t_{k-i}$ 0. Variables in the set A, which was returnd by SUPPRESS_G, suppress all the terms in $Term(g)$. In the following steps of L, the region of positive examples is restricted to the one consisting of positive examples satisfying the condition $v_A = 0$.

While there remain at least two terms each of which covers reasonably large region of the region obtained by the restriction mentioned above, another variable that suppresses at least one of such terms is chosen. Then the variable chosen is added to the set A of variables to restrict the region further by putting them 0. This is done until there remains only one term with associated region not too small (procedure SUPPRESS_F).

Restricting the region of positive examples by putting appopreately chosen variables 0 is the key idea behind our algorithm. The restriction reduces the problem of learning k-term MDNF to that of learning MDNF with less terms and less variables. In general this makes the problem easy. In fact, in the case where there remains only one term with associated region not too small, it is easy to identify the term (procedure MAKE_TERM).

The whole algorithm produces the formula that approximates the target after iterate these steps at most k times.

The learning algorithm L is given in Figure 3. Procedure SUPPRESS_G is given in Figure 4, procedure SUPPRESS_F is in Figure 5, and procedure MAKE_TERM is in Figure 6. '$A_1 \times A_2 \times \cdots \times A_i$' denotes the cartesian product set of i sets $A_1, A_2, \ldots, A_i$. Let **0** and **1** denote the Boolean formulae corresponding to the constant Boolean functions. (See Figure 3, Figure 4 and Figure 6.)

```
begin
    $g := \mathbf{0}$ ; $A := \emptyset$ ; $p_c := 1$ ;
    for $i := k$ down_to 1 do
        begin
            if $i \neq k$ then
                begin
                    $S :=$ EXAMPLES($\frac{32}{\varepsilon} \ln \frac{\delta}{4(k-1)}$, TRUE, , ) ;
                    if FREQ($S$, $\varepsilon/2$, $\varepsilon$, $g(v)=0$) = "low" then break_for_loop ;
                    $A :=$ SUPPRESS_G($g$, $i$, $\delta/4(k-1)$) ;
                    $p_c := \frac{1}{2} \cdot \frac{\varepsilon}{i2^{|A|+1}}$ ;
                end
            ($A$, $j$, $p_c$) := SUPPRESS_F($A$, $i$, $\delta/4k$, $p_c$) ;
            $t :=$ MAKE_TERM($A$, $j$, $\delta/4k$, $p_c$) ;
            $g := g \vee t$ ;
        end
    return $g$ ;
end.
```

Figure 3: Algorithm L for learning k-term MDNF

```
procedure  SUPPRESS_G($g$, $i$, $\delta$)
begin
    $\mathcal{A} := Var(t_1) \times \cdots \times Var(t_{k-i})$ ;
    $S :=$ EXAMPLES($\frac{64i2^{k-i}}{\epsilon} \ln \frac{|\mathcal{A}|}{\delta}$, TRUE, , ) ;
    foreach  $A$  in  $\mathcal{A}$  do
        if  FREQ($S$, $\frac{1}{2} \cdot \frac{\epsilon}{i2^{|A|+1}}$, $\frac{\epsilon}{i2^{|A|+1}}$, $v_A = 0$) = "high"  then return  $A$ ;
end
```

Figure 4: Procedure SUPPRESS_G

```
procedure  SUPPRESS_F($A$, $i$, $\delta$, $p_c$)
    $j := i$ ;
    while  $j > 1$  do
        begin
            $V_1 := V_2 := \emptyset$ ;  $A^c := \{x_1, \ldots, x_n\} - A$ ;
            $S :=$ EXAMPLES($64j \ln \frac{4(i-1)|A^c|}{\delta}$, $v_A = 0$, $p_c$, $\delta/4(i-1)$) ;
            foreach  $x_l$  in  $A^c$  do
                if  FREQ($S$, $\frac{1}{2} \cdot \frac{1}{2j}$, $\frac{1}{2j}$, $v_l = 0$) = "high"
                    then  $V_1 := V_1 \cup \{x_l\}$ ;
            $S :=$ EXAMPLES($24j^2 2^{2j} \ln \frac{4(i-1)|A^c|}{\delta}$, $v_A = 0$, $p_c$, $\delta/4(i-1)$) ;
            foreach  $x_l$  in  $A^c$  do
                if  FREQ($S$, $\frac{1}{2}$, $\frac{1}{2}(1 + \frac{1}{2} \cdot \frac{1}{j2^{j-1}})$, $v_l = 1$) = "high"
                    then  $V_2 := V_2 \cup \{x_l\}$ ;
            if  $V_1 \cap V_2 = \emptyset$  then break_while_loop
            else
                begin
                    take one $x_l$ from $V_1 \cap V_2$ and let $A := A \cup \{x_l\}$ ;
                    $p_c := p_c \cdot \frac{1}{2} \cdot \frac{1}{2j}$ ;
                    $j := j - 1$ ;
                end
        end
    return  ($A$, $j$, $p_c$) ;
end
```

Figure 5: Procedure SUPPRESS_F

```
procedure MAKE_TERM($A$, $j$, $\delta$, $p_c$)
begin
    $t := 1$ ; $A^c := \{x_1, \ldots, x_n\} - A$ ;
    $S :=$ EXAMPLES($24j^2 2^{2j}(1+\frac{1}{j2^j})\ln\frac{2|A^c|}{\delta}$, $v_A=0$, $p_c$, $\delta/2$) ;
    foreach $x_l$ in $A^c$ do
        if FREQ($S$, $\frac{1}{2}(1+\frac{1}{2}\cdot\frac{1}{j2^{j-1}})$, $\frac{1}{2}(1+\frac{1}{j2^{j-1}})$, $v_l=1$) = "high"
            then $t := t \wedge x_l$ ;
    return $t$ ;
end
```

Figure 6: Procedure MAKE_TERM

4 Correctness of the learning algorithm

Many of the statements in this section are involved with Lemma 2.5. As is shown in the remark after the lemma, we simply claim that the fact that FREQ returnes "low" implies $\Pr[E(v)] < p$, and that the fact that FREQ returnes "high" implies $\Pr[E(v)] > p - \alpha$ without saying the phrase "with high probability". Also we simply claim that EXAMPLES produces the desired sequence, omitting the same phrase.

Fact 4.1 $\left(1-\frac{\delta}{m}\right)^m \geq 1-\delta$ for any $m \geq 1$, $0 \leq \delta \leq 1$.

Lemma 4.2 Let f be a non-redundunt formula in k-term MDNF and t be any term in *Term*(f). Let v be a random positive example of f. For any variable x_i in *Var*(t), $\Pr[v_i = 1] \geq \frac{1}{2}\left(1 + \frac{1}{2^{k-1}} \cdot \Pr[t(v) = 1]\right)$.

Proof. Let T be the set of terms t' in *Term*(f) such that $x_i \in$ *Var*(t'). Let f' be the disjunction of terms in T and f'' be the disjunction of terms in *Term*$(f) - T$. Then,

$$\begin{aligned}\Pr[v_i = 1] &= \Pr[f''(v) = 1 \text{ and } v_i = 1] + \Pr[f''(v) = 0 \text{ and } f'(v) = 1]\\ &\geq \tfrac{1}{2}(\Pr[f''(v) = 1] + \Pr[f''(v) = 0 \text{ and } f'(v) = 1] + \Pr[f''(v) = 0 \text{ and } t(v) = 1])\\ &= \tfrac{1}{2}(1 + \Pr[f''(v) = 0 \text{ and } t(v) = 1]).\end{aligned}$$

From Fact 3.1, there exists a set A such that $|A| \leq k-1$, $A \cap$ *Var*$(t) = \emptyset$ and that $v_A=0$ implies $f''(v) = 0$. Therefore, $\Pr[f''(v) = 0 \text{ and } t(v) = 1] \geq \Pr[v_A=0 \text{ and } t(v) = 1]$ $= \frac{1}{2^{|A|}}\Pr[t(v) = 1] \geq \frac{1}{2^{k-1}}\Pr[t(v) = 1]$. □

Proposition 4.3 Let A be the set returnd by the procedure SUPPRESS_G(g, i, δ). Then $\Pr[v_A=0] > \frac{1}{2} \cdot \frac{\varepsilon}{i2^{|A|+1}}$.

Proof. If the procedure SUPPRESS_G is called by the learning algorithm L, it is guaranteed that $\Pr[g(v) = 0] > \varepsilon/2$. This fact is easily verified from the remark after Lemma 2.5.

From $\Pr[g(v)=0] > \varepsilon/2$ and the fact that there exist at most i terms in $\mathit{Term}(f) - \mathit{Term}(g)$, there exists a term t in $\mathit{Term}(f)-\mathit{Term}(g)$ such that $\Pr[t(v)=1 \text{ and } g(v)=0] > \frac{1}{i}\cdot\frac{\varepsilon}{2}$, which implies $\Pr[t(v)=1] > \frac{\varepsilon}{2i}$. Therefore, from Fact 3.1 there exists a set A in $\mathcal{A}$ such that $A \cap \mathit{Var}(t) = \emptyset$ and $\Pr[t(v)=1 \text{ and } v_A{=}0] > \frac{1}{2^{|A|}}\cdot\frac{\varepsilon}{2i}$. Therefore, it follows that there exists an A such that $\Pr[v_A{=}0] > \frac{\varepsilon}{i2^{|A|+1}}$.

From $|A| \le k-i$, it is easy to see that $\frac{64i2^{k-i}}{\varepsilon}\ln\frac{|A|}{\delta}$, the length of the sequence S produced by the procedure EXAMPLES in SUPPRESS_G, satisfies the condition (2-1). Therefore, from Lemma 2.5, there exists an A such that the procedure FREQ returns "high". On the other hand, if $\Pr[v_A{=}0] < \frac{1}{2}\cdot\frac{\varepsilon}{i2^{|A|+1}}$ holds, the procedure FREQ returns "low". Thus, it is concluded that SUPPRESS_G correctly returnes A such that $\Pr[v_A{=}0] > \frac{1}{2}\cdot\frac{\varepsilon}{i2^{|A|+1}}$. □

Note that all terms in $\mathit{Term}(g)$ are suppressed by the valiables in A which was returnd by SUPPRESS_G.

Proposition 4.4 Let p_c and A be as in the procedure SUPPRESS_F. For any A during the procedure SUPPRESS_F, $\Pr[v_A{=}0] \ge p_c$.

Proof. From Proposition 4.3, $\Pr[v_A{=}0]$ is greater than p_c when the procedure SUPPRESS_F was called from L.

Now assume that $\Pr[v_A{=}0] \ge p_c$ at the beginning of the while loop for some iteration step. Let x_l be the variable added to A in this iteration step of SUPPRESS_F.

From Proposition 2.4, the procedure EXAMPLES, which is called first in this iteration, produces a sequence S composed of $64j\ln\frac{4(i-1)|A^c|}{\delta}$ positive examples satisfying the condition $v_A{=}0$ under the assumption. Each vector in S was drawn uniformly from the positive examples satisfying $v_A{=}0$. It is easy to see that above length of the sequence satisfies the condition (2-1). Thus, from Lemma 2.5, the condition for variables in V_1 that FREQ(S, $\frac{1}{2}\cdot\frac{1}{2j}$, $\frac{1}{2j}$, $v_l{=}0$) returns "high" implies $\Pr[v_l=0 \mid v_A{=}0] > \frac{1}{2}\cdot\frac{1}{2j}$. Therefore we have $\Pr\left[v_{A\cup\{x_l\}}{=}0\right] = \Pr[v_l=0 \text{ and } v_A{=}0] = \Pr[v_l=0 \mid v_A{=}0]\cdot\Pr[v_A{=}0] > \frac{1}{2}\cdot\frac{1}{2j}\cdot\Pr[v_A{=}0]$. And p_c is set to $p_c\cdot\frac{1}{2}\cdot\frac{1}{2j}$ whenever x_l is added to A. Therefore the assumption is true for the next iteration step.

Thus, the assumption is true for all iteration step, and the proposition follows. □

From Proposition 4.3 and Proposition 4.4, it follows that $\Pr[v_A{=}0] = \Omega(\varepsilon)$ during the whole algorithm L. Also note that Proposition 4.4 implies that there always remains at least one term in $\mathit{Term}(f) - \mathit{Term}(g)$ that is not suppressed by the valiables in A.

Proposition 4.5 Let x_l be the variable added to A in the procedure SUPPRESS_F. There exists some term in $\mathit{Term}(f)$ that is suppressed by x_l but not by the variables in A.

Proof. From Proposition 2.4 and Proposition 4.4, the procedure EXAMPLES, which is called second in each iteration, produces a sequence S of $24j^2 2^{2j}\ln\frac{4(i-1)|A^c|}{\delta}$ positive examples satisfying the condition $v_A{=}0$. The first bound of the condition (2-1) of Lemma 2.5 becomes $16j^2 2^{2j}(1+\frac{1}{j2^j})\ln\frac{4(i-1)|A^c|}{\delta} \ge 16j^2 2^{2j}(1+\frac{1}{2})\ln\frac{4(i-1)|A^c|}{\delta} = 24j^2 2^{2j}\ln\frac{4(i-1)|A^c|}{\delta}$

for $j \geq 1$, and the right most term is equal to the second bound of (2-1). Thus, above length of the sequence satisfies the condition (2-1). Therefore, from Lemma 2.5, the condition for variables in V_2 that FREQ(S, $\frac{1}{2}$, $\frac{1}{2}(1+\frac{1}{2}\cdot\frac{1}{j2^{j-1}})$, $v_l\!=\!1$) returns "high" implies $\Pr[v_l = 1 \mid v_A\!=\!0] > 1/2$, the proposition follows. □

Lemma 4.6 Let f be a target formulae. The procedure MAKE_TERM called from the learning algorithm L returns a term in $\mathit{Term}(f) - \mathit{Term}(g)$.

Proof. Let j and A be those in the procedure MAKE_TERM.

Assume that $j = 1$. From Proposition 4.5, variables in A suppresses $k-1$ terms in $\mathit{Term}(f)$. Therefore, there remains just one term t not suppressed by variables in A. Therefore, $\Pr[v_l = 1 \mid v_A\!=\!0]$ is equal to 1 if x_l is in $\mathit{Var}(t)$, and $1/2$ otherwise. Thus, as the following argument shows, the remaining term t is produced correctly in MAKE_TERM.

Now assume that $j > 1$. In this case, the conditions for values returned by FREQ in the procedure SUPPRESS_F are not satisfied: For any variable x_l in $A^c = \{x_1, \ldots, x_n\} - A$, $\Pr[v_l = 0 \mid v_A\!=\!0] < \frac{1}{2j}$ or $\Pr[v_l = 1 \mid v_A\!=\!0] < \frac{1}{2}\left(1+\frac{1}{2}\cdot\frac{1}{j2^{j-1}}\right)$. Since there remain at most j terms not suppressed by the variables in A, there exists a term t in $\mathit{Term}(f)$ $-\mathit{Term}(g)$ such that $\Pr[t(v) = 1 \mid v_A\!=\!0] > \frac{1}{j}$. Therefore, by Lemma 4.2, $\Pr[v_l = 1 \mid v_A\!=\!0]$ $> \frac{1}{2}\left(1+\frac{1}{2^{j-1}}\cdot\frac{1}{j}\right)$ for any x_l in $\mathit{Var}(t)$. On the other hand, for any x_l in $A^c - \mathit{Var}(t)$, we have $\Pr[v_l = 0 \text{ and } t(v) = 1 \mid v_A\!=\!0] = \Pr[v_l = 0 \mid t(v) = 1 \text{ and } v_A\!=\!0] \cdot \Pr[t(v) = 1 \mid v_A\!=\!0]$ $> \frac{1}{2}\cdot\frac{1}{j}$, which implies $\Pr[v_l = 0 \mid v_A\!=\!0] > \frac{1}{2j}$. Therefore, since the first condition for x_l chosen in SUPPRESS_F is satisfied, the second condition is not satisfied: $\Pr[v_l = 1 \mid v_A\!=\!0]$ $< \frac{1}{2}\left(1+\frac{1}{2}\cdot\frac{1}{j2^{j-1}}\right)$ for any x_l in $A^c - \mathit{Var}(t)$.

From Proposition 2.4 and Proposition 4.4, the procedure EXAMPLES, which is called in MAKE_TERM, produces a sequence of $24j^2 2^{2j}(1+\frac{1}{j2^j})\ln\frac{2|A^c|}{\delta}$ positive examples satisfying the condition $v_A\!=\!0$. The first bound of the condition (2-1) of Lemma 2.5 becomes $16j^2 2^{2j}(1+\frac{1}{j2^{j-1}})\ln\frac{4(i-1)|A^c|}{\delta}$, and the second bound becomes $24j^2 2^{2j}\left(1+\frac{1}{j2^j}\right)\ln\frac{4(i-1)|A^c|}{\delta}$. It is easy to see that the second bound is greater than the first one for $j \geq 1$. Thus, above length of the sequence satisfies the condition (2-1). Therefore, from Lemma 2.5, it is concluded that the term t in $\mathit{Term}(f) - \mathit{Term}(g)$ is produced in MAKE_TERM. □

Theorem 4.7 The learning algorithm L for k-term MDNF learns k-term MDNF in time $O\left(\frac{n^k}{\varepsilon}(\ln n + \ln\frac{1}{\delta})\right)$, using $O\left(\frac{1}{\varepsilon}(\ln n + \ln\frac{1}{\delta})\right)$ positive examples.

Proof. Correctness of the algorithm is immediate from Lemma 4.6. All that remain is to estimate the number of examples and the time required, and the confidence parameters.

Procedure SUPPRESS_G(g, i, δ) calls procedure FREQ at most $|\mathcal{A}|$ times. Each FREQ is called with confidence parameter $\delta/|\mathcal{A}|$. Thus, from Fact 4.1, SUPPRESS_G(g, i, δ) achieves $1-\delta$ confidence.

Procedure SUPPRESS_F(A, i, δ, p_c) iterates the steps inside the while loop at most $i-1$ times. In each iteration, procedure EXAMPLES is called twice with confidence parameter $\delta/4(i-1)$, and procedure FREQ is called $2|A^c|$ times with confidence parameter $\delta/4(i-1)|A^c|$. Therefore, from Fact 4.1, each iteration has confidence $(1-\frac{\delta}{4(i-1)})^2 \cdot (1-\frac{\delta}{4(i-1)|A^c|})^{2|A^c|} \geq (1-\frac{\delta}{2(i-1)})^2 \geq 1-\delta/(i-1)$. Therefore, SUPPRESS_F($A$, i, δ, p_c) achieves $1-\delta$ confidence.

Procedure MAKE_TERM(A, j, δ, p_c) calls EXAMPLES once with confidence parameter $\delta/2$, and calls FREQ $|A^c|$ times with confidence parameter $\delta/2|A^c|$. Therefore, the same argument as above shows that MAKE_TERM(A, j, δ, p_c) achieves $1-\delta$ confidence.

Algorithm L itself calls FREQ at most $k-1$ times with confidence parameter $\delta/4(k-1)$, calls SUPPRESS_G at most $k-1$ times with confidence parameter $\delta/4(k-1)$, calls SUPPRESS_F at most k times with confidence parameter $\delta/4k$, and calls MAKE_TERM at most k times with confidence parameter $\delta/4k$. From Fact 4.1, it is concluded that the whole algorithm L achieves $1-\delta$ confidence.

Procedure EXAMPLES called by L itself calls POS $\mathcal{O}(\frac{1}{\varepsilon}\ln\frac{1}{\delta})$ times. From $1 \leq i \leq k$ and $|\mathcal{A}| \leq n^{k-1}$, EXAMPLES called by SUPPRESS_G calls POS $\mathcal{O}\big(\frac{1}{\varepsilon}(\ln n + \ln\frac{1}{\delta})\big)$ times. From $|A^c| < n$, $1 \leq i, j \leq k$ and the fact that $p_c = \Omega(\varepsilon)$ during the whole algorithm L, it is easy to see that each EXAMPLES called by SUPPRESS_F and MAKE_TERM calls POS $\mathcal{O}\big(\frac{1}{\varepsilon}(\ln n + \ln\frac{1}{\delta})\big)$ times. Since the numbers of iterations in L and SUPPRESS_F are both at most k, it is concluded that the total number of examples required is $\mathcal{O}\big(\frac{1}{\varepsilon}(\ln n + \ln\frac{1}{\delta})\big)$.

When the procedure SUPPRESS_G(g, i, δ) is called from L, at most $|\mathcal{A}| \leq n^{k-i}$ sets A must be tested via $\mathcal{O}\big(\frac{1}{\varepsilon}(\ln n + \ln\frac{1}{\delta})\big)$ positive examples. This part dominates the time required over the whole algorithm. Therefore, it is easy to see that the total time required is $\mathcal{O}\big(\frac{n^k}{\varepsilon}(\ln n + \ln\frac{1}{\delta})\big)$.

Completed the whole proof of Theorem 4.7. $\square$

5 Learning k-term MDNF in the presence of errors

In this section, we consider the learnability of k-term MDNF in the presence of noise in the examples, adopting the *malicious error model* which is introduced in [V85] and studied in [KL].

$\mathrm{POS}^{\mathrm{M}}_{\beta}(f)$ is an oracle with following features: With probability $1-\beta$, it behaves the same as POS(f) and, with probability β, it produces an arbitrary answer possibly chosen maliciously by an adversary. $\mathrm{POS}^{\mathrm{M}}_{\beta}(f)$ is called POS(f) *with malicious error rate* β, and simply written as $\mathrm{POS}^{\mathrm{M}}_{\beta}$ when no confusion arises. Without loss of generality, $0 \leq \beta < 1/2$ is assumed. Since we are concerned with only positive examples, we do not need to consider NEG(f) with malicious error. By replacing POS with $\mathrm{POS}^{\mathrm{M}}_{\beta}$ in the definition of learnability in section 2, we have the definition of learnability from $\mathrm{POS}_\beta^{\mathrm{M}}$.

$\Pr[E(v) : \mathrm{POS}]$ denotes the probalility of an event $E(v)$ where v is given by POS, and $\Pr\left[E(v) : \mathrm{POS}_\beta^{\mathrm{M}}\right]$ denotes the probalility of the event $E(v)$ where v is given by $\mathrm{POS}_\beta^{\mathrm{M}}$.

Fact 5.1 Let r, s be such that $0 \le r, s \le 1$. If $\Pr[E(v) : \mathrm{POS}] \ge r$ then $\Pr\left[E(v) : \mathrm{POS}_\beta^{\mathrm{M}}\right] \ge (1-\beta)r$, and if $\Pr[E(v) : \mathrm{POS}] \le s$ then $\Pr\left[E(v) : \mathrm{POS}_\beta^{\mathrm{M}}\right] \le (1-\beta)s + \beta$.

Following lemma is analogous to Lemma 2.5. It tells that the procedure FREQ can be used to estimete the probability of an event in the case of occuring errors as well.

Lemma 5.2 Let p, α, δ be as in the Lemma 2.5. Let S be a sequence of m vectors each of which is given independently by $\mathrm{POS}_\beta^{\mathrm{M}}$, and let m be such that $m \ge \frac{48p}{\alpha^2}\ln\frac{1}{\delta}$. Let β, β_0 satisfy the following condition (5-2).

$$0 \le \beta \le \beta_0 \le \alpha/4 \tag{5-2}$$

Let $p' = (1-\beta_0)p$ and $\alpha' = \alpha - (1+\alpha)\beta_0$. If $\Pr[E(v) : \mathrm{POS}] \le p - \alpha$ then FREQ(S, $p' - \alpha'$, p', δ) returns "high" with probability at most δ, and if $\Pr[E(v) : \mathrm{POS}] \ge p$ then it returns "low" with probability at most δ.

Proof. Using Fact 5.1 and an argument analogous to the proof of Lemma 2.5, the lemma can be proved. Details are left to the readers. □

When we attempt to estimete the conditional probability $\Pr[E(v)|C(v) : \mathrm{POS}]$, the situation is slightly different. Let S be a sequence of vectors produced by EXAMPLES(m, C(v), p_c, δ), where POS is replaced with $\mathrm{POS}_\beta^{\mathrm{M}}$ in the procedure EXAMPLES. Each vectors in S can be seen as a vector that is produced independently by an oracle which produces positive examples satisfying the condition $C(v)$ uniformly with malicious error rate β'. From the assumption that $\Pr[C(v) : \mathrm{POS}] \ge p_c$, $\beta < 1/2$, and Fact 5.1, it follows that $\beta' = \beta/\Pr\left[C(v) : \mathrm{POS}_\beta^{\mathrm{M}}\right] \le \beta/(1-\beta)p_c < 2\beta/p_c$. Therefore, replacing β with β' in Lemma 5.2, we have that if β and β_0 satisfy the followong condition (5-3), then Lemma 5.2 can be used to estimete the conditional probability as well.

$$0 \le \beta \le \beta_0 \le \alpha p_c/8 \tag{5-3}$$

Theorem 5.3 There is a constant $0 < c_k < 1$ such that k-term MDNF is learnable from $\mathrm{POS}_\beta^{\mathrm{M}}$ for $\beta = c_k\varepsilon$.

Proof. Applying the modification suggested by Lemma 5.2, it is easy to modify the learning algorithm L to the learning algorithm L* that learns k-term MDNF in the presence of errors. And it can be seen that in the modified algorithm, right most terms in the condition (5-2) and (5-3) are both $\Omega(\varepsilon)$. Thus, Lemma 5.2 and Lemma 4.7 imply the theorem. Details are left to the readers. □

Acknowledgement

We would like to thank Les Valiant for pointing us [KMP].

References

[GM] Qian Ping GU and Akira Maruoka, "Leaning Monotone Boolean Functions by Uniformly Distributed Examples", submitted.

[H] D. Haussler, "Quantifying the inductive bias in concept learning", extended abstract in *Proc. AAAI '86*, Philadelphia, PA., 1986, pp.485–489.

[KL] M. Kearns and M. Li, "Learning in the presence of malicious errors", In *proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pp.267–280, 1988.

[KLPV87a] M. Kearns, M. Li, L. Pitt and L. G. Valiant, "Recent Results on Boolean Concept Learning", *Proc. 4th Int. Workshop on Machine Learning*, Morgan Kaufmann, Los Altos, CA., 1987, pp.337–352.

[KLPV87b] M. Kerns, 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*, pp.285–295, 1987.

[KMP] L. Kucera, A. Marchetti-Spaccamela and M. Protasi, "On the learnability of DNF formulae", *Lecture Notes in Computer Science*, 317:347–361, 1988.

[PV] L. Pitt and L. G. Valiant, "Computational limitations on learning from examples", *Tech. Rept. TR-05-86, Harvard University*, 1986.

[V84] L. G. Valiant, "A theory of the learnable", *Communications of the ACM*, 27(11), 1984, pp.1134–42.

[V85] L. G. Valiant, "Learning disjunctions of conjunctions", *Proc. 9th IJCAI*, Los Angeles, CA, 1985.