Machine Learning, 5, 151-164 (1990)
© 1990 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

Polynomial Time Learnability of Simple Deterministic Languages

HIROKI ISHIZAKA (ISHIZAKA@ICOT.JP)
ICOT Research Center, 21F, Mita Kokusai Bldg., 1-4-28 Mita, Minato-ku, Tokyo 108, Japan

Abstract. This paper is concerned with the problem of learning simple deterministic languages. The algorithm described in this paper is based on the theory of model inference given by Shapiro. In our setting, however, nonterminal membership queries, except for the start symbol, are not permitted. Extended equivalence queries are used instead. Nonterminals that are necessary for a correct grammar and their intended models are introduced automatically. We give an algorithm that, for any simple deterministic language L, outputs a grammar G in 2-standard form, such that $L = L(G)$, using membership queries and extended equivalence queries. We also show that the algorithm runs in time polynomial in the length of the longest counterexample and the number of nonterminals in a minimal grammar for L.

Keywords. Language learning, Polynomial time learning, Simple deterministic languages, Generating nonterminals.

1. Introduction

We consider the problem of learning simple deterministic languages using membership queries and extended equivalence queries. A simple deterministic language (SDL) is a language that is accepted by a 1-state deterministic push-down automaton by empty store. The class of SDLs is a proper sub-class of deterministic languages. The SDLs may also be characterized as the languages that are generated by context-free grammars in a special form of Greibach normal form, called simple deterministic grammars (SDGs).

Angluin (1987a) shows that the class of k-bounded context-free grammars is learnable in polynomial time using membership queries, nonterminal membership queries and equivalence queries. The algorithm described in this paper is based on her algorithm. Both algorithms are essentially based on the theory of model inference given by Shapiro (1983). Our setting, however, differs from Angluin's and Shapiro's in the types of queries that are available to the learning algorithm. That is, the algorithm is allowed to use membership queries but not nonterminal membership queries. This difference leads to the problem of introducing new nonterminals that are not observed in interactions between the teacher and the learner.

This relates to the problem of introducing theoretical terms in the learning of first order theories from facts. Recently, there have been several approaches to this problem (Banerji, 1988; Muggleton & Buntine, 1988). However, in settings where the algorithm learns not only a concept but also a language for describing the concept, it becomes difficult to ensure the convergence of a learning process. Of course, if the concepts are described by a sufficiently restricted language, then we can expect to have an algorithm that learns the concept even in such a setting. This paper presents one such learning algorithm.

Another feature of our setting is that the algorithm is allowed to use extended equivalence queries. The equivalence query defined in (Angluin, 1988) is allowed to conjecture only elements of the original hypothesis space. For example, if the target class[1] of learning is a set of concept representations $R = \{r_1, r_2, \ldots\}$, then any equivalence query made by the learning algorithm must be with some r_i from R. We lift this restriction in this paper. In particular, the learning algorithm described in this paper is allowed to make an equivalence query conjecturing any grammar in 2-standard form; not necessarily simple deterministic. Hence, each intermediate hypothesis conjectured by an extended equivalence query might define a general context-free language.

Yokomori (1988) gives another algorithm for learning SDLs in polynomial time. Our setting also differs from his, as will be described in Section 4. Berman and Roos (1987) show that the class of deterministic one-counter languages is learnable in polynomial time using membership queries and equivalence queries. Although the class of one-counter languages is incomparable with the class of SDLs[2], it is interesting that both are classes with decidable equivalence problems.

2. Preliminaries

We will give some basic notions and the notation needed in this paper. We follow Angluin (1987a) and Yokomori (1988) wherever possible. The algorithm and its presentation parallel those of Angluin's result on learning k-bounded context-free grammars.

2.1. Context-free grammars and languages

An *alphabet* is a finite non-empty set of distinct symbols. For a given alphabet X, the set of all finite strings of symbols from X is denoted X^*. The empty string is denoted ϵ. X^+ denotes the set $X^* - \{\epsilon\}$. For a string x, $|x|$ denotes the length of x. If S is a finite set, then $|S|$ denotes the cardinality of S.

Let Σ be an alphabet. A *language* L over Σ is a subset of Σ^*. For a string x in Σ^* and a language L over Σ, let $\bar{x}L = \{y \mid xy \in L\}$ ($L\bar{x} = \{y \mid yx \in L\}$). The set $\bar{x}L$ ($L\bar{x}$) is called the *left(right)-derivative of L with respect to x*. For a string $x = a_1a_2 \cdots a_n$, $Pre_i(x)$ denotes the string $a_1a_2 \cdots a_i$, and $Suf_i(x)$ denotes the string $a_{i+1}a_{i+2} \cdots a_n$.

A *context-free grammar* (CFG) is a 4-tuple $G = (N, \Sigma, P, S)$, where N is an alphabet of *nonterminals*, Σ is an alphabet of *terminals* such that $N \cap \Sigma = \emptyset$, $S \in N$ is the *start symbol*, and P is a finite set of *production rules* of the form $A \rightarrow \alpha$, where $A \in N$, $\alpha \in (N \cup \Sigma)^*$. A CFG G is in *Greibach normal form* if and only if each production rule of G is of the form $A \rightarrow a\alpha$, where $A \in N$, $a \in \Sigma$ and $\alpha \in N^*$. Note that, in this paper, we consider only ϵ-free grammars and languages. A CFG G is said to be in m-standard form if G is in Greibach normal form and, for each production $A \rightarrow a\alpha$ of G, $|\alpha| \leq m$. The *size* of a grammar G is the sum of $|N|$, $|\Sigma|$, $|P|$, and the sum of the lengths of the right-hand sides of all the productions in P.

For $\beta, \gamma \in (N \cup \Sigma)^*$, binary relation $\Rightarrow$ is defined as follows: $\beta \Rightarrow \gamma$ if and only if there exist $\delta_1, \delta_2 \in (N \cup \Sigma)^*$ and a production rule $A \rightarrow \alpha \in P$ such that $\beta = \delta_1 A \delta_2$

and $\gamma = \delta_1\alpha\delta_2$. A *derivation from* β *to* γ is a finite sequence of strings $\beta = \beta_0, \beta_1, \ldots, \beta_n = \gamma$ such that, for each i, $\beta_i \Rightarrow \beta_{i+1}$. If there is a derivation from β to γ, then we denote it by $\beta \Rightarrow^* \gamma$, that is, the relation $\Rightarrow^*$ is the reflexive, transitive closure of $\Rightarrow$. In each step of a derivation, if the left-most nonterminal occurrence in β_i is replaced, then such a derivation is said to be a *left-most derivation* of γ from β. In what follows, unless otherwise stated, $\beta \Rightarrow^* \gamma$ denotes a left-most derivation of γ from β.

The *language of a nonterminal* A, denoted $L(A)$, is the set of all $x \in \Sigma^*$ such that $A \Rightarrow^* x$. Similarly, for $\alpha \in N^*$, $L(\alpha)$ denotes the set of all $x \in \Sigma^*$ such that $\alpha \Rightarrow^* x$. (To emphasize the grammar being used, we sometimes use the subscript G, for example, $S \Rightarrow_G x$ or $L_G(A)$.) The *language of a grammar* G, denoted $L(G)$, is just $L(S)$, where S is the start symbol of G. A language L is called *context-free* if there exists a CFG G such that $L = L(G)$.

A *derivation tree* T of a grammar $G = (N, \Sigma, P, S)$ is a tree such that each internal node of T is labeled with an element of N, each leaf of T is labeled with an element of Σ and, for each internal node labeled with $A \in N$, there exists a production $A \rightarrow \alpha$ in P, where $\alpha \in (N \cup \Sigma)^*$ is the concatenation of the labels of its children in left-to-right order. Let T be a derivation tree of a grammar. The root label of T is denoted by $rt(T)$. The *frontier* of T, denoted by $fr(T)$, is the concatenation of the labels of its leaves in left-to-right order. A derivation tree T illustrates a derivation from $rt(T) \in N$ to $fr(T) \in \Sigma^*$.

2.2. SDG and SDL

A context-free grammar in Greibach normal form G is *simple deterministic* if the following condition holds: for any $A \in N$, $a \in \Sigma$, $\alpha, \beta \in N^*$, if there exist productions $A \rightarrow a\alpha$ and $A \rightarrow a\beta$ in P, then $\alpha = \beta$. A language L is *simple deterministic* if there exists an SDG G such that $L(G) = L$.

For example, the grammar $G = (\{S, A, B, C\}, \{a, b\}, P, S)$, where

$$P = \{S \rightarrow aA,\ A \rightarrow b,\ A \rightarrow aB,\ B \rightarrow aBC,\ B \rightarrow bC,\ C \rightarrow b\},$$

is one of the SDGs that generate an SDL $\{a^m b^m | 1 \leq m\}$.

The following propositions (see, for example, (Harrison, 1979)) provide properties of SDGs and SDLs that are useful for our purpose.

PROPOSITION 1. For any SDG $G = (N, \Sigma, P, S)$, G is unambiguous, that is, for any $w \in L(G)$, there is a unique left-most derivation of w from S.

PROPOSITION 2. Let $G = (N, \Sigma, P, S)$ be an SDG. For any $A \in N$, $x \in \Sigma^+$ and $\alpha \in N^*$, if there exists a derivation $A \Rightarrow^* x\alpha$, then $L(\alpha) = \bar{x}L(A)$.

PROPOSITION 3. Let $G = (N, \Sigma, P, S)$ be an SDG. For any $A \in N$, $L(A)$ is prefix-free, that is, if $x \in L(A)$, then, for any $y \in \Sigma^+$, $xy \notin L(A)$.

Proposition 4. For any SDG G, there exists an equivalent SDG G' that is in 2-standard form, that is, there exists an SDG $G' = (N', \Sigma, P', S)$ such that

1. $L(G) = L(G')$;
2. Each production in P' is of one of the following forms: $A \to a$, $A \to aB$, $A \to aBC$, where $A, B, C \in N'$, $a \in \Sigma$.

Proposition 4 allows us to consider only (ϵ-free) context-free grammars in 2-standard form as the hypotheses of our learning algorithm.

We will analyze the complexity of our learning algorithm on two types of complexity measures: one is the length of the given example strings and the other is the number of nonterminals of a *minimal SDG* for the target language. Let L be an SDL. A *minimal SDG for L* is an SDG $G = (N, \Sigma, P, S)$ in 2-standard form satisfying the following conditions:

1. $L(G) = L$;
2. For any SDG $G' = (N', \Sigma, P', S')$ in 2-standard form such that $L(G') = L$, $|N| \leq |N'|$.

2.3. Models and incorrectness/correctness

Our algorithm for learning SDLs is based on Shapiro's (1983) model inference algorithm and Angluin's (1987a) learning algorithm for k-bounded CFGs. The most important component of these algorithms is the diagnosis routine. The diagnosis routine identifies an incorrect clause in a hypothesized theory which wrongly implies a negative instance. In our present context, we need to clarify the notion of incorrectness of a production in a grammar. In order to do this, we introduce some model theoretic notions for grammars.

Let $G = (N, \Sigma, P, S)$ be a context-free grammar. For each nonterminal $A \in N$, a *model* of A, denoted $M(A)$, is a subset of Σ^+. A *model M* for the grammar G consists of a model of each nonterminal.

$$M = \{M(A_1), M(A_2), \ldots, M(A_{|N|})\}.$$

A *replacement* is a finite tuple (possibly empty) of pairs of a terminal string $y_i \in \Sigma^*$ and a nonterminal $A_i \in N$:

$$\langle (y_1, A_1), \ldots, (y_n, A_n) \rangle.$$

Let $\rho = \langle (y_1, A_1), \ldots, (y_n, A_n) \rangle$ be a replacement and β be a string in $(N \cup \Sigma)^*$. ρ is *compatible* with β if and only if there are finite strings $x_0, \ldots, x_n \in \Sigma^*$ such that $\beta = x_0 A_1 x_1 A_2 \cdots A_n x_n$. If ρ is compatible with β, then the *instance* of β by ρ, denoted $\rho[\beta]$, is the terminal string obtained from β by replacing each occurrence of A_i in β by the terminal string y_i. An empty replacement ρ is compatible with any terminal string x and $\rho[x] = x$.

Let M be a model for a grammar G. A production $A \rightarrow \alpha$ is *incorrect* for M if and only if there exists a replacement $\rho = \langle (y_1, A_1), \ldots, (y_n, A_n) \rangle$ that is compatible with α such that, for each i, $y_i \in M(A_i)$[3], but $\rho[\alpha] \notin M(A)$. A production is *correct* for M if and only if it is not incorrect for M.

For example, consider the SDG given in the previous section. Let M be a model such that, for each nonterminal $X \in N$, $M(X) = L(X)$, that is

$$\begin{aligned} M = \{\, & M(S) = \{a^m b^m \mid 1 \leq m\},\ M(A) = \{a^{m-1} b^m \mid 1 \leq m\}, \\ & M(B) = \{a^{m-2} b^m \mid 2 \leq m\},\ M(C) = \{b\}\}. \end{aligned}$$

Then, a production $A \rightarrow aBC$ is incorrect for M, because there exists a replacement $\rho = \langle (bb, B), (b, C) \rangle$ that is compatible with aBC such that $bb \in M(B)$, $b \in M(C)$, but the string $\rho[aBC] = abbb$ is not in $M(A)$. More generally, we have the following.

Proposition 5. Let $G = (N, \Sigma, P, S)$ be a CFG and M be a model for G such that, for each nonterminal $A \in N$, $M(A) = L(A)$. Then every production in P is correct for M.

2.4. Types of queries

Let L be the target SDL to be learned by our learning algorithm. We assume there is a teacher who knows L and can answer the following two types of queries:

A *membership query* proposes a string $x \in \Sigma^+$ and asks whether $x \in L$. The reply is either *yes* or *no*.

An *extended equivalence query* conjectures a grammar G in 2-standard form and asks whether $L = L(G)$. The reply is either *yes* or *no*. If it is *no*, then a *counterexample* is also provided. A counterexample is a string x in the symmetric difference of L and $L(G)$. If $x \in L - L(G)$, x is called a *positive* counterexample, and if $x \in L(G) - L$, x is called a *negative* counterexample. The choice of a counterexample is assumed to be arbitrary.

Note the difference between the extended equivalence query and the equivalence query defined in (Angluin, 1988). The equivalence query is only allowed to conjecture members of the target class. Thus, in learning SDLs, any hypothesis conjectured by the algorithm would have to be a grammar generating an SDL. In contrast, the hypothesis conjectured by an extended equivalence query does not have to generate an SDL.

A teacher who answers equivalence queries and membership queries was called a *minimally adequate teacher* (Angluin, 1987b). We call a teacher who answers extended equivalence queries and membership queries an *extended minimally adequate teacher.*

The notion of the extended equivalence query corresponds to the notion, in the context of the PAC-learning model, of learning the target class R *in terms of* the class of representations R', not necessarily identical to R (see, for example, (Pitt & Warmuth, 1988)). Informally, R is said to be PAC-learnable in terms of R' if there exists a polynomial time algorithm A such that for any target concept (description) $r \in R$, if A is given randomly chosen examples of r, A outputs, with high probability, a concept (description) $r' \in R'$ that approximates the target concept r. In our setting, R corresponds to the class of SDGs (or SDLs) and R' corresponds to the class of CFGs in 2-standard form. In general, such a relaxation of the learnability criterion enriches the learnable classes of concepts. For example, the class of k-term DNFs is not PAC-learnable in terms of itself unless **RP** = **NP**, but the class is learnable in terms of the class of k-CNFs. For the result given in this paper, however, the learnability of SDGs in terms of itself (the learnability of SDGs from a minimally adequate teacher) is still open.

3. The learning algorithm

Let L be the unknown SDL to be learned by the algorithm and $G_0 = (N_0, \Sigma, P_0, S)$ be a minimal SDG for L. We assume that the terminal alphabet Σ and start symbol S are known to the learning algorithm, but that $N - \{S\}$, the set of nonterminals except S, and P, the set of productions, are unknown.

The main result of this paper is as follows.

THEOREM 6. There is an algorithm that, for any SDL L, outputs a grammar G in 2-standard form such that $L(G) = L$ using extended equivalence queries and membership queries. Moreover, at any point during the run, the time used by the algorithm to that point is bounded by some polynomial in $|N_0|$, the number of nonterminals of a minimal SDG for L, and the length of the longest counterexample returned by any equivalence query seen to that point.

Note that the grammar learned by the algorithm may not be an SDG. The grammar is simply in 2-standard form.

In this section, we will describe our learning algorithm. We begin by giving an informal description of the algorithm, omitting details concerning the exact nature of the new nonterminals and their intended models, which will later be made precise.

3.1. An outline of the algorithm

First, the algorithm initializes N to $\{S\}$, and P to the set of all productions containing S as the only nonterminal. As a model M for G, we initially consider $\{M(S) = L\}$. Models for any other nonterminals introduced by the algorithm will be defined in Section 3.3. Then the algorithm iterates the following loop: An extended equivalence query is made, conjecturing G. If the reply is *yes*, then the algorithm outputs G and halts. Otherwise, a counterexample w is returned. The algorithm tries to find a derivation tree T of G such that $rt(T) = S$ and $fr(T) = w$. If it exists, that is, when w is a negative counterexample, the algorithm diagnoses G on T and finds an incorrect production for M. The incorrect production is removed from P. Otherwise, that is, when w is a positive counterexample, new nonterminals are introduced and all new productions constructed from them are added to P.

The Learning Algorithm

```
Given: An extended minimally adequate teacher for $L$ and a terminal alphabet $\Sigma$.
Output: A grammar $G = (N, \Sigma, P, S)$ in 2-standard form such that $L(G) = L$.
Procedure:

    $N := \{S\}.\ P := \{S \to aSS,\ S \to aS,\ S \to a | a \in \Sigma\}.\ G := (N, \Sigma, P, S).$

repeat
  Make an extended equivalence query with $G$.
  If the reply is a positive counterexample, then
    introduce new nonterminals with their models.
    Put all candidate productions into $P$.
  Else if the reply is a negative counterexample, then
    diagnose $G$.
    Remove the incorrect production returned by the diagnosis routine from $P$.
until the reply is yes.
Output $G$.
```

In this paper, we assume a parsing sub-procedure that runs in time polynomial in the size of a grammar G and $|w|$, for example, Angluin's (1987a) parsing procedure[4]. In the following two subsections, we describe the diagnosis routine and how new nonterminals and productions are generated. Then, in the third subsection, we show the correctness and characterize the complexity of the entire algorithm.

3.2. Diagnosing an incorrect hypothesis

The diagnosis routine finds an incorrect production for M on an input derivation tree T of G such that $fr(T) \notin M(rt(T))$. It is essentially a special case of the *contradiction backtracing algorithm* given by Shapiro (1983).

For a given input deviation tree T, the diagnosis routine considers, in turn, each child of the root of T. If the child is labeled with a nonterminal and T' is the sub-tree rooted at the child, then the diagnosis routine inquires whether $fr(T') \in M(rt(T'))$. If $fr(T') \notin M(rt(T'))$, then it calls itself recursively with T'. Otherwise, it goes on to the next child of the root of T. If there is no nonterminal child such that $fr(T') \notin M(rt(T'))$, for the sub-tree T' rooted at the child, then the diagnosis routine returns the production $rt(T) \to \alpha \in P$, where α is the concatenation of the labels of the children of the root of T in left-to-right order.

For example, consider the derivation tree for a negative counterexample *abbb* in Figure 1. Initially, $abbb \notin M(S) = L$. First, the child labeled with A generating the string bb is considered. The diagnosis routine inquires whether $bb \in M(A)$. If $bb \notin M(A)$, then it calls itself recursively with the sub-tree rooted at the child. If $bb \in M(A)$, then it goes to the next child labeled with B and makes a similar inquiry. If $b \notin M(B)$, then it returns the production $B \to b$. Otherwise, it returns the production $S \to aAB$.

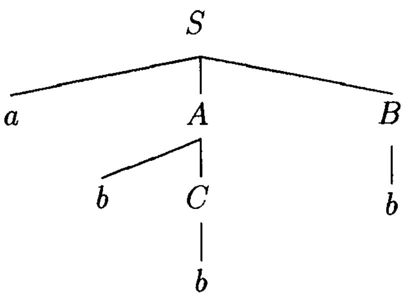

Figure 1. An example of an input to the diagnosis routine.

In (Angluin, 1987a), such a diagnosis is made through nonterminal membership queries of the type "$bb \in L(A)$?" In our approach, it is performed through membership queries only. The next section shows how to introduce new nonterminals and replace nonterminal membership queries by membership queries.

LEMMA 7. Suppose that the diagnosis routine is given as its input a derivation tree T of G such that $fr(T) \notin M(rt(T))$. Then it returns a production in P that is incorrect for M.

Proof. Since each recursive call is with a proper sub-tree of its input derivation tree, the diagnosis routine must eventually terminate and output some production in P (since each sub-tree is also a derivation tree of G, the output production is a member of P).

Let $A \rightarrow \alpha$ be the returned production. For each nonterminal occurrence X in the production, let T_X be the sub-tree of T that is rooted at the corresponding node labeled with X in T. From the input condition, it holds that $fr(T_A) \notin M(A)$. If α contains no nonterminal, then the empty replacement ρ satisfies $\rho[\alpha] = \alpha = fr(T_A) \notin M(A)$. Otherwise, from the termination condition of the procedure, for each B_i appearing in α, $fr(T_{B_i}) \in M(B_i)$. Thus there exists a replacement $\rho = \langle(fr(T_{B_1}), B_1), \ldots, (fr(T_{B_n}), B_n)\rangle$ that is compatible with α such that, for each i, $fr(T_{B_i}) \in M(B_i)$, but $\rho[\alpha] = fr(T_A) \notin M(A)$. So $A \rightarrow \alpha$ is incorrect for M.

Note that, at the initial call to the diagnosis routine, the input derivation tree T is for a negative counterexample w. Since $fr(T) = w \notin L = M(S) = M(rt(T))$, the input condition is satisfied initially.

3.3. Generating nonterminals and productions

The key idea of the nonterminal-generating routine has its roots in an extension of a model described in (Ishizaka, 1989).

First, we show an important feature of SDGs for describing the nonterminal generating routine.

LEMMA 8. Let $G = (N, \Sigma, P, S)$ be an SDG. Suppose that $A \Rightarrow^* rB\alpha$ for $A, B \in N$, $\alpha \in N^*$, $r \in \Sigma^+$, and that t is a string in $L(\alpha)$ such that $Suf_j(t) \notin L(\alpha)$ for any j $(1 \le j \le |t| - 1)$ (if $\alpha = \epsilon$ then $t = \epsilon$). Then, for any $x \in \Sigma^+$, $x \in L(B)$ if and only if (i) $rxt \in L(A)$ and (ii) $rPre_i(x)t \notin L(A)$ for any $i(1 \le i \le |x| - 1)$.

Proof. Suppose $x \in L(B)$. Then $A \Rightarrow^* rB\alpha \Rightarrow^* rx\alpha \Rightarrow^* rxt$. Thus, $rxt \in L(A)$. Since $L(B)$ is prefix-free, $Pre_i(x) \notin L(B)$ for any $i(1 \le i \le |x| - 1)$. Hence, if $rPre_i(x)t \in L(A)$, that is, $Pre_i(x)t \in \bar{r}L(A) = L(B\alpha)$, then there exists $j(1 \le j \le |t| - 1)$ such that $Pre_i(x)Pre_j(t) \in L(B)$ and $Suf_j(t) \in L(\alpha)$. This contradicts the fact that $Suf_j(t) \notin L(\alpha)$ for any $j(1 \le j \le |t| - 1)$. Thus, $rPre_i(x)t \notin L(A)$ for any $i(1 \le i \le |x| - 1)$.

Conversely, assume that (i) and (ii) hold. From (i), it follows that $xt \in \bar{r}L(A) = L(B\alpha)$. Since there is no proper suffix of t in $L(\alpha)$, there exists $j(1 \le j \le |x|)$ such that $Pre_j(x) \in L(B)$ and $Suf_j(x)t \in L(\alpha)$. On the other hand, from (ii), $Pre_i(x)t \notin L(B\alpha)$ for any $i(1 \le i \le |x| - 1)$. Hence, for any $i(1 \le i \le |x| - 1)$, $Pre_i(x) \notin L(B)$. Thus, $j = |x|$. This shows that $Pre_{|x|}(x) = x \in L(B)$.

In the learning algorithm, new nonterminals are introduced whenever there is a positive counterexample w. The nonterminal-generating routine constructs nonterminals with their appropriate models from w.

Let w be a positive counterexample such that $|w| \ge 2$. *Nonterminals generated from a positive counterexample* w, denoted $N(w)$, are defined as follows:

$$N(w) = \{(r, s, t) | r, s \in \Sigma^+, t \in \Sigma^* \text{ and } rst = w\}.$$

For each triple $(r, s, t) \in N(w)$, let $\varphi(r, s, t)$ be the shortest suffix of t in $\overline{rs}L$, that is,

$$\varphi(r, s, t) = Suf_i(t) \text{ where } i = \max_{0 \le j \le |t| - 1} \{j \mid Suf_j(t) \in \overline{rs}L\}.$$

The intended model of each nonterminal in $N(w)$ is defined as follows. For each triple $(r, s, t) \in N(w)$, define

$$M((r, s, t)) = \{x \in \Sigma^+ \mid rx\varphi(r, s, t) \in L \text{ and } rPre_i(x)\varphi(r, s, t) \notin L \text{ for any } i(1 \le i \le |x| - 1)\}.$$

Let w be a newly given positive counterexample at a stage of learning. Then N is set to $N \cup N(w)$. Let $P_{N(w)}$ be a set of all productions in 2-standard form constructed from N that have never appeared in P, that is, for each $a \in \Sigma$, $P_{N(w)}$ contains productions $A \to a\alpha$ such that $A\alpha \in N^+$, $|\alpha| \le 2$ and $A\alpha$ contains at least one element of $N(w)$. Then P is set to $P \cup P_{N(w)}$. Note that, at any point during the learning, P contains at most $|N| \times |\Sigma| \times (|N| + 1)^2$ productions for N generated by the algorithm to that point.

LEMMA 9. Let N be the set of known nonterminals. Suppose that w is a new positive counterexample. Then the time required for generating nonterminals and computing new productions is bounded by a nondecreasing polynomial in $|N|$ and $|w|$.

Proof. There are at most $|w|(|w| - 1)/2$ nonterminals in $N(w)$, and $N(w)$ is computable in time polynomial in $|w|$. For each (r, s, t)in $N(w)$, the string $\varphi(r, s, t)$ is computed by making at most $|t|$ membership queries with the strings $rsSuf_j(t)$ $(0 \leq j \leq |t| - 1)$. Moreover the set $P_{N(w)}$ is computable in time polynomial in $|N|$ and $|N(w)|$. These facts prove the lemma.

LEMMA 10. Let L be an SDL, w be a string in L, and $G = (N, \Sigma, P, S)$ be an SDG such that $L(G) = L$. For any $A \in N - \{S\}$ that appears in the derivation $S \Rightarrow^* w$, there exists a nonterminal $(r, s, t) \in N(w)$ such that $L(A) = M((r, s, t))$.

Proof. Suppose that $S \Rightarrow^* rA\alpha \Rightarrow^* rs\alpha \Rightarrow^* rst = w$. Then, from the definition of $N(w)$, the triple (r, s, t) is in $N(w)$. (Since G is an SDG and $A \neq S$, neither r nor s is ϵ.) Since $L(S) = L(G) = L$, by Proposition 2, $L(\alpha) = \overline{rs}L(S) = \overline{rs}L$. By the definition of $\varphi(r, s, t)$, $\varphi(r, s, t) \in L(\alpha)$ and $Suf_j(\varphi(r, s, t)) \notin L(\alpha)$ for any $j(1 \leq j \leq |\varphi(r, s, t)| - 1)$. Hence, by Lemma 8 and the definition of $M((r, s, t))$, $L(A) = M((r, s, t))$.

The above lemma ensures that if the learning algorithm is given a positive counterexample w, then it can make all nonterminals with appropriate models that are necessary for generating w. As a result, nonterminal membership queries used by Angluin's (1987a) or Shapiro's (1983) algorithm can be replaced by membership queries. For any $x \in \Sigma^*$ and $A \in N(w)$, the diagnosis routine can accomplish each inquiry as to whether $x \in M(A)$ by making $|x|$ membership queries.

3.4. *Correctness and complexity*

In what follows, let $G = (N, \Sigma, P, S)$ be the current hypothesis of the algorithm and

$$M = \{M(S), M((r_1, s_1, t_1)), \ldots, M((r_{|N|-1}, s_{|N|-1}, t_{|N|-1}))\}$$

be the model for G defined in the previous section.

LEMMA 11. At any point during the learning, the time required by the diagnosis routine on an input derivation tree for a negative counterexample w is bounded by a nondecreasing polynomial in $|w|$ and ℓ_p, where ℓ_p is the length of the longest positive counterexample returned by any equivalence query seen to that point.

Proof. Since G is in 2-standard form, there are at most $|w|$ occurrences of nonterminals in the derivation tree. Thus, the number of inquiries made by the diagnosis routine is at most $|w|$. For each inquiry as to whether $x \in M(A)$ or not, if $A = S$, then only one membership query "$x \in L$?" is made. Otherwise, that is, if $A = (r, s, t)$, the routine makes at most $|x|$ membership queries "$rPre_i(x)\varphi(r, s, t) \in L$?" for $1 \leq i \leq |x|$. Since x is a substring of w, the total number of queries made in a diagnosing process is at most $|w|^2$. Since the main operations performed in the diagnosis routine are forming strings $rPre_i(x)\varphi(r, s, t)$ and making membership queries, it is clear that the claim of the lemma holds.

LEMMA 12. Let $G_0 = (N_0, \Sigma, P_0, S)$ be a minimal SDG for the target language L. The total number of given positive counterexamples is bounded by $|N_0|$.

Proof. Let w_n be the n-th positive counterexample given to the learning algorithm. We define $N_0(w_n)$ and $P_0(w_n)$ as follows:

$$N_0(w_n) = \{A \in N_0 \mid \exists u \in \Sigma^+, \exists \alpha \in N^*, S \Rightarrow^*_{G_0} uA\alpha \Rightarrow^*_{G_0} w_n\},$$

$$P_0(w_n) = \{A \to a\alpha \in P_0 \mid a \in \Sigma, A\alpha \in (\bigcup_{i=1}^{n} N_0(w_i))^+\}.$$

When w_n is given, the learning algorithm computes $N(w_n)$ and sets N to $N \cup N(w_n)$. Then it computes all new candidate productions and adds them to P as described in the previous section.

By Lemma 10, for each nonterminal $A \in N_0(w_n)$, there exists a nonterminal $A' \in N(w_n)$ such that $L(A) = M(A')$. Under this correspondence of A and A', for every production in $P_0(w_n)$, a corresponding production is added to P at least once. By Proposition 5, these corresponding productions are correct for M. Since correct productions are never removed from P, whenever the n + 1st positive counterexample is given, there exists at least one nonterminal $A \in N_0$ such that

$$A \in N_0(w_{n+1}) \text{ and } A \notin \bigcup_{i=1}^{n} N_0(w_i).$$

Thus, the number of given positive counterexamples is at most $|N_0|$.

LEMMA 13. At any point during the learning, the number of nonterminals introduced by the learning algorithm is bounded by $|N_0|\ell_p(\ell_p - 1)/2$, where ℓ_p is the length of the longest positive counterexample returned by any equivalence query seen to that point.

Proof. For each positive counterexample w_i, $|N(w_i)|$ is at most $|w_i|(|w_i| - 1)/2$ as stated in the previous section. By Lemma 12, the total number of nonterminals introduced by the algorithm is bounded by $|N_0|\ell_p(\ell_p - 1)/2$.

Proof of Theorem 6. From the method of introducing new productions and Lemma 13, the total number m of productions introduced into P is at most

$$m = \frac{|N_0|\ell_p(\ell_p - 1)}{2} \times |\Sigma| \times \left(\frac{|N_0|\ell_p(\ell_p - 1)}{2} + 1 \right)^2.$$

By Lemma 7, for each given negative counterexample, at least one incorrect production is found and it is removed from P. With Lemma 12, this implies that, after given at most $|N_0|$ positive counterexamples and at most m negative ones, the learning algorithm outputs a grammar G such that $L(G) = L$.

By Lemma 13, at any point during the learning, the size of G is bounded by a nondecreasing polynomial in $|N_0|$ and ℓ, where ℓ is the length of the longest counterexample given to that point. From the assumption on the parsing sub-procedure, the algorithm can determine whether a given counterexample is positive or negative in time polynomial in $|N_0|$ and ℓ. The total number of given counterexamples is at most $|N_0| + m$. With Lemma 9 and Lemma 11, this proves the claim, made in Theorem 6, on the complexity of the learning algorithm.

4. Conclusion

We have considered the problem of learning SDLs. The main idea presented in this paper was a method of introducing necessary nonterminals with their appropriate models (or interpretations). The problem of introducing new, unobserved sub-concepts that are useful for representing a target concept is one of the most important and difficult problems in machine learning. Although there have been several approaches to this problem (for example, Banerji, 1988; Muggleton et al. 1988), it seems that none of the solutions proposed to date is satisfactory. Our result presented in this paper is no exception: it suffers from the limitation that the class shown as being learnable is too restricted for many practical applications. The method for introducing nonterminals given in this paper depends heavily upon the structural properties of SDGs used in the proof of Lemma 8. For example, the uniqueness of a left-most derivation is one of them. So the method is not applicable to (at least) the target class containing an ambiguous grammar. In future, we would like to find more general and practical solutions to this challenging problem.

The efficiency of the algorithm given in this paper is also not optimal. As shown in the proof of Theorem 6, it is ensured that the algorithm runs in time polynomial in $|N_0|$ and ℓ. The polynomial has a rather high degree. The polynomial is larger than, at least, the size of the largest hypothesis, $O(|N_0|^3\ell^6)$. If we can set each intermediate hypothetical grammar to an SDG, we may be able to decrease the degree. While a grammar G in 2-standard form has, in the worst case, $|N| \times |\Sigma| \times (|N| + 1)^2$ productions, an SDG G has at most $|N| \times |\Sigma|$ productions. Since the operation performed most frequently by the algorithm is the parsing of each given counterexample on each hypothesis G, this reduction in size of each hypothetical grammar will decrease the complexity of the learning algorithm. Obviously, such a restriction on hypothetical grammars also results in the development of an algorithm that produces an SDG as its output using normal equivalence queries and membership queries. The efficient learnability of SDLs from a minimally adequate teacher is still open.

Yokomori (1988) gives another algorithm for learning SDLs in polynomial time. His algorithm conjectures only SDGs. In his setting, however, a very powerful teacher is assumed. The teacher can answer the following two types of queries: prefix membership queries and derivatives equivalence queries. A prefix membership query is an extension of the membership query. A derivatives equivalence query proposes two pairs of strings (u_1, w_1), (u_2, w_2) and asks whether $\overline{u_1}L\overline{w_1} = \overline{u_2}L\overline{w_2}$, where L is the target language. It is clear that derivatives equivalence queries can be used, in our algorithm, to test whether two candidate nonterminals are identical. For example, for two nonterminals (u_1, v_1, w_1) and (u_2, v_2, w_2), if $\overline{u_1}L\overline{w_1} = \overline{u_2}L\overline{w_2}$, then they are identical. Thus, the number of nonterminals generated by our algorithm will be reduced. The relationship between the power of the teacher and the efficiency of the learning algorithm remains an interesting open question.

Acknowledgments

I wish to thank Dr. K. Furukawa and Dr. R. Hasegawa for their continuous support and advice. I am also deeply grateful to Dr. T. Yokomori for his many valuable comments. Mr. Y. Sakakibara pointed out my misunderstanding about equivalence queries in an earlier draft of this paper. Dr. K. Fuchi, the director of ICOT Research Center, gave me the opportunity to conduct this research into the Fifth Generation Computer Systems Project. Finally, I wish to thank three anonymous referees for a lot of very constructive and instructive comments.

Notes

1. *The target class* is a class of representations that define the class of concepts to be learned by the algorithm (see, for example, (Pitt, 1989)).
2. For example, the language $\{\{a^nb^n \mid n \geq 1\}c\}^+$ is deterministic one-counter, but not simple deterministic. On the other hand, the language $\{a^mb^nca^nb^m \mid m, n \geq 1\}$ is simple deterministic, but not deterministic one-counter.
3. When α has no nonterminal, then this condition is not necessary.
4. Since G is in 2-standard form, Lemma 3 and Lemma 4 in (Angluin, 1987a) hold. In fact, the procedure returns a parse-DAG (directed acyclic graph) instead of a derivation tree. Our discussion, however, is not affected by the difference.

References

Angluin, D. (1987a). *Learning k-bounded context-free grammars*. (Technical Report YALEU/DCS/RR-557). New Haven, CT: Yale University, Department of Computer Science.

Angluin, D. (1987b). Learning regular sets from queries and counterexamples. *Information and Computation*, *75*, 87-106.

Angluin, D. (1988). Queries and concept learning. *Machine Learning*, *2*, 319-342.

Banerji, R.B. (1988). Learning theories in a subset of a polyadic logic. *Proceedings of the First Workshop on Computational Learning Theory* (pp. 281-295). San Mateo, CA: Morgan Kaufmann.

Berman, P. & Roos, R. (1987). Learning one-counter languages in polynomial time. *Proceedings of the Twenty-Eighth IEEE Symposium on Foundations of Computer Science* (pp. 61-67). New York: The Institute of Electrical and Electronics Engineers.

Harrison, M.A. (1979). *Introduction to Formal Language Theory*. Reading, MA: Addison-Wesley.

Ishizaka, H. (1989). Inductive inference of regular languages based on model inference. *International Journal of Computer Mathematics*, *27*, 67-83.

Muggleton, S. & Buntine, W. (1988). Machine invention of first-order predicates by inverting resolution. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 339–352). San Mateo, CA: Morgan Kaufmann.

Pitt, L. (1989). Inductive inference, dfas, and computational complexity. *Proceedings of the AII-89 Workshop on Analogical and Inductive Inference; Lecture Notes in Artificial Intelligence 397,* (pp. 18–44). Heidelberg: Springer-Verlag.

Pitt, L. & Warmuth, M.K. (1988). Prediction preserving reducibility. *J. Comput. Sys. Sci.* To appear.

Shapiro, E.Y. (1983). *Algorithmic program debugging.* Cambridge, MA: MIT Press.

Yokomori, T. (1988). Learning simple languages in polynomial time. *Proceedings of SIG-FAI* (pp. 21-30). Tokyo, Japan: Japanese Society for Artificial Intelligence.