A Note on Learning Decision Lists

Jorge Castro

Report LSI-95-2-R
A note on learning decision lists

Jorge Castro
Dept. Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Email: castro@lsi.upc.es

January 13, 1995

Abstract
We show an algorithm that learns decision lists via equivalence queries, provided that a set $G$ including all terms of the target list is given. The algorithm runs in time polynomial in the cardinality of $G$. From this learning algorithm, we prove that log $n$-decision lists—the class of decision lists such that all their terms have low Kolmogorov complexity—are simple pac-learnable.

1 Introduction

In Valiant's original model of learning, "pac-learning" [7], one has to learn a target concept with high probability, in polynomial time, from a polynomial number of examples, within a certain error, under all probability distributions on the examples. This last requirement, to learn under all distributions, is a strong one. Many concept classes are not known to be polynomially learnable or known not to be polynomially learnable if RP $\neq$ NP, although some such concept classes are polynomially learnable under some fixed distributions. However, learning under one fixed distribution may be too restrictive to be useful.

Li and Vitányi proposed in [3] the simple pac-learning model with the aim to get a compromise between practicability—in the sense that more concept classes are learnable—and usefulness, understood as the property that guarantees that the concepts are learnable under a wide and interesting class of distributions. Roughly speaking, simple pac-learning model replaces the condition of learning under all distributions of Valiant's original model by the request of learning under all simple distributions, provided the sample is given according to the "universal" distribution (see next paragraph).

Simple distributions form a wide class that properly include all enumerable ones. Specifically, they are those multiplicatively dominated by the universal enumerable distribution $m$. This distribution assigns high probabilities to low Kolmogorov complexity examples. As we will see later, these properties of $m$ have nice consequences with regard to pac-learning under simple distributions.

With the new model, Li and Vitányi [3] developed a theory of learning for simple concepts—concepts with low Kolmogorov complexity—that intuitively should be polynomially learnable. In fact, they showed several examples that strengthen this intuition.
A decision list \( f \) over \( n \) Boolean variables is a sequence of terms \( f_1 \ldots f_t \), such that each term \( f_i \), for \( i = 1, \ldots, t - 1 \), is a pair \( (f_m, f_b) \) where \( f_m \) is a monomial and \( f_b \) is either 0 or 1. The last term \( f_t \) is always \( (1, f_b) \). The value of a decision list \( f \) on a setting of the \( n \) Boolean variables is defined to be \( f_b \), where \( i \) is the least number such that \( f_m \) is satisfied by the assignment. The class of \( k \)-decision lists consists of all decision lists \( f \) for which each monomial \( f_m \) has at most \( k \) literals. Rivest gave in [5] an algorithm for learning \( k \)-decision lists, for each constant \( k \).

We will show here an algorithm for exact learning, via equivalence queries, of decision lists. It works under the assumption that a set \( G \), including all terms of the target list, is known. This algorithm runs in time polynomial in the cardinality of \( G \). From this learning algorithm, we can solve the simple pac-learning problem for \( \log n \)-decision lists, cited as open in [3] (see also page 308 of [4]). \( \log n \)-decision lists is the class of all decisions lists \( f \) over \( n \) Boolean variables such that each term \( f_i \) of \( f \) has Kolmogorov complexity \( O(\log n) \).

2 Preliminaries

We follow sections 5.5 and 5.6 of [4] and section 2 of [3]. In this paper we work with a discrete sample space \( S \). The elements of \( S \) are called examples. A concept \( c \) is a subset of \( S \). Abusing notation, we use \( c \) and the characteristic function \( f : S \rightarrow \{0, 1\} \) of \( c \) interchangeably for the concept \( c \subseteq S \) and for the syntactic representation of \( c \). For a concept \( c \), we denote by \( l(c) \) the minimum length of the representations of \( c \). A concept class \( C \) is a set of concepts. Fixed \( f \) as the target concept, a learning algorithm draws examples from the sample space \( S \) according to a fixed but unknown probability distribution \( P \). Each example \( e \) comes with a label that shows the value of \( f(e) \).

**Definition 2.1** A concept class \( C \) is pac-learnable iff there exists a learning algorithm \( A \) such that, for each \( f \in C \) and \( \epsilon \) \((0 < \epsilon < 1)\), algorithm \( A \) always halts within time and number of examples \( p(l(f), 1/\epsilon) \), for some polynomial \( p \), and outputs a concept \( h \in C \) which satisfies

\[
\text{Prob}(P(h \neq f) < \epsilon) > 1 - \epsilon.
\]

Note that the pac-learning model requires that the algorithm learns under all distributions (the distribution \( P \) in the definition is unknown). We can also define, in a natural way, pac-learning under a class \( \Delta \) of distributions: simply requiring definition 2.1 only on distributions from \( \Delta \).

It can be proved that there exists an universal enumerable distribution, denoted by \( m \), that multiplicatively dominates each enumerable distribution. It can be shown that

\[
m(x) = 2^{-K(x) + O(1)},
\]

where \( K(x) \) denotes the Kolmogorov complexity of \( x \). The universal distribution has many important properties. Under \( m \), easily describable objects have high probability, and complex objects have low probability.
Definition 2.2  A distribution $P$ is simple iff it is multiplicatively dominated by the universal distribution. That is, there exists a constant $c$, such that for all $x$, 

$$cm(x) \geq P(x).$$

It can be shown that simple distributions properly include enumerable ones and that there is a distribution which is not simple. The following theorem relates learning under simple distributions with learning under the universal distribution $m$.

Theorem 2.3 ([3]) A concept class $C$ is pac-learnable under the universal distribution $m$, iff it is pac-learnable under simple distributions, provided that in the learning phase the set of examples is drawn according to $m$.

This theorem says roughly that a concept class is polynomially learnable under $m$ if and only if it is polynomially learnable under each simple distribution. In [3] it is shown how to exploit this completeness theorem to obtain new learning algorithms. Now, we define

Definition 2.4  A concept class $C$ is simple pac-learnable iff it is pac-learnable under $m$.

We assume the reader knows the exact learning or learning via queries models (see [1] and [2]). In particular, we will work with the learning via equivalence queries model. Roughly speaking, in this model the learning algorithm has to produce in polynomial time a representation of the target concept, doing a polynomial number of equivalence queries. An equivalence query asks whether a representation given by the algorithm is a representation of the target concept. Equivalence queries are answered by a teacher affirmatively – when the asked representation is a representation of the target concept – or with a counterexample that shows the difference.

A pac-learning algorithm can be obtained from a learning via equivalence queries algorithm by means of a now standard transformation (see [1] and [2]):

Theorem 2.5  Let $C$ be a concept class. If $C$ is learnable via equivalence queries then $C$ is pac-learnable.

In the next section we will use this theorem to show a simple pac-learning algorithm for $\log n$-decision lists. We will proceed as follows. First we will give an algorithm that, assuming a set $G$ including all terms of the target list is known, learns the target list doing equivalence queries in time polynomial in the cardinality of $G$. After that, we will show that if the target list is a $\log n$-decision list and we draw a polynomial number of examples according to the universal distribution $m$, with high probability we can obtain a set $G$ of polynomial size that includes all terms of the target list. From these facts we may conclude that $\log n$-decision lists are simple pac-learnable.
3 Learning decision lists

Let \( f = f_1 \ldots f_t \) be a decision list with \( f_i = \langle f m_i, f b_i \rangle \), for \( i = 1, \ldots, t - 1 \), and \( f_t = \langle 1, f b_1 \rangle \). Let \( G = \{ g_1, \ldots, g_s \} \) be a set of terms with \( g_i = \langle g m_i, g b_i \rangle \) for \( i = 1, \ldots, s \), and let us assume that \( \{ f_1, \ldots, f_t \} \subseteq G \).

We consider the following algorithm \texttt{Learn_Dlist} that knowing the set \( G \), tries to learn \( f \) doing equivalence queries. Here \( E \) denotes a set of labeled examples according to \( f \), obtained from previous queries.

\[
\text{function } \texttt{Learn_Dlist} (G) \text{ ret } g : \text{decision_list} \\
E := \emptyset \\
g := \texttt{Group_list}(G, E) \\
\text{Ask the equivalence query } g = f? \\
\text{while the teacher does not reply "yes" do} \\
\quad \text{Let } e \text{ be the new counterexample} \\
\quad \text{Let } g_i = \langle g m_i, g b_i \rangle \text{ be the first term of } g \text{ such that } g m_i(e) = 1 \\
\quad E := E \cup \langle e, g b_i \rangle \\
\quad g := \texttt{Group_list}(G, E) \\
\quad \text{Ask the equivalence query } g = f? \\
\text{endwhile} \\
\text{return } g \\
\text{endfunction}
\]

The core of the algorithm is the function \texttt{Group_list}. Intuitively, this function returns a decision list \( g \) by grouping all terms of \( G \) according to their behaviors on \( E \). Given a labeled example \( \langle e, b \rangle \) of \( E \) and a term \( g_i \) of \( G \), we say that \( \langle e, b \rangle \) is a consistency (resp. an inconsistency) of \( g_i \) iff \( g m_i(e) = 1 \) and \( g b_i = b \) (\( g b_i = \overline{b} \)). Each term \( g_i \) of \( G \) appears in \( g \) placed in the first group of terms with the following property: each inconsistency of \( g_i \) is a consistency of some term in a previous group. The function \texttt{Group_list} is formally defined as follows (here ++ denotes list concatenation):

\[
\text{function } \texttt{Group_list} (G', E) \text{ ret } g : \text{decision_list} \\
\text{for all } i = 1, \ldots, s \text{ do} \\
\quad \text{Let } C_i \text{ be the set of labeled examples } \langle e, b \rangle \in E \text{ such that } g m_i(e) = 1 \text{ and } g b_i = b \\
\quad (* \text{ let us call } C_i \text{ the consistent set of the term } g_i *) \\
\quad \text{Let } I_i \text{ be the set of labeled examples } \langle e, b \rangle \in E \text{ such that } g m_i(e) = 1 \text{ and } g b_i = \overline{b} \\
\quad (* \text{ let us call } I_i \text{ the inconsistent set of the term } g_i *) \\
\text{endfor} \\
\text{g := null_list} \\
\text{while } G' \neq \emptyset \text{ do} \\
\quad \text{Let } \texttt{Newgroup} \text{ be the list of terms of } G' \text{ such that their inconsistent}
sets are empty. It does not matter the order in the list
\[ g := g++\text{Newgroup} \]
\[ G' := G' - \{\text{terms of Newgroup}\} \]
for all \( g_i \in G' \) do
  for all \( g_j \in \text{Newgroup} \) do
    \( I_i := I_i - (I_i \cap C_j) \)
endfor
endwhile
return \( g \)
endfunction

Assuming that \text{Group.list} halts, each time that it is called it generates a new list \( g \), in such a way that \( g = gr(g,1)++\ldots++gr(g,k) \), where \( gr(g,1) \) - the first group of \( g \) - is a list of terms having no inconsistencies in \( E \), and in general, \( gr(g,i+1) \) is a list of terms whose inconsistencies in \( E \) are included in the union of the consistent sets of terms in previous groups. Now, we are ready to prove the following lemma

**Lemma 3.1** Assuming that \( G \) includes all terms of \( f \), the algorithm \text{Learn.Dlist} halts and learns \( f \) in time polynomial in \( s \), where \( s \) is the cardinality of \( G \).

**Proof** We have just to prove that \text{Learn.Dlist} runs in time polynomial in \( s \). We proceed by steps.

**Fact 1.** The function \text{Group.list} halts in time polynomial in \( s \) and the cardinality of \( E \). We show that \text{Newgroup} is always non empty. At the beginning of the process of \text{Group.list}, the set \( G' \) is \( G \) and includes all terms \( f_1, \ldots, f_t \) of the target list \( f \). The first group \( gr(g,1) \) is always nonempty because \( f_1 \) does not have inconsistencies. Let us suppose that \( gr(g,i) \) is the last group constructed and \( G' \neq \emptyset \). Either, all terms of \( f \) are already in a group, or there exists a term of \( f \) in \( G' \). In the first case, the term \( f_t = (1, fb_i) \) of \( f \) is in a group. As each example of \( E \) is either consistent or inconsistent for \( f_t \), we may conclude that all inconsistent sets of terms in \( G' \) have already been erased. So, \text{Newgroup} is \( G' \).

In the second case, let \( f_{\sigma} \), the first term of \( f \) that belongs to \( G' \). As \( E \) is a set of examples of \( f \), the inconsistent set of \( f_{\sigma} \), is empty because all its inconsistencies in \( E \) are included in the union of the consistent sets of \( f_1, \ldots, f_{\sigma-1} \) that, by hypothesis, belong to previous groups. So, \( f_{\sigma} \) belongs to \text{Newgroup}.

**Fact 2.** The decision list \( g \) returned by \text{Group.list} has at most \( t + 1 \) groups of terms, where \( t \) is the number of terms of \( f \).

Using ideas from above it can be shown the following: \( f_1 \in gr(g,1) \), so \( f_2 \in gr(g,1) \cup gr(g,2) \) and, in general, \( f_i \in gr(g,j) \) and \( j \leq i \). So, \( f_t = (1, fb_i) \in gr(g,j) \) and \( j \leq t \). Now the result is clear.
Fact 3. Let $g_i$ be a term of $G$ and let $gr(g, \sigma_i)$ be the group of $g_i$ before processing \texttt{Group.List}. Let $gr(g, \mu_i)$ be the group of $g_i$ after processing \texttt{Group.List}. It holds that $\sigma_i \leq \mu_i$.

When a new counterexample is put in $E$ the inconsistent set of $g_i$ cannot decrease. A contradiction appears if we suppose $\sigma_i > \mu_i$ for some $1 \leq i \leq s$.

Fact 4. The term $g_i$ has to change of group by processing \texttt{Group.List}.

Let us suppose that $g_i \in gr(g, i)$ before processing \texttt{Group.List}. Note that the last counterexample will be put in the inconsistent set of $g_i$ and will not be put in the consistent sets of terms in $gr(g, j)$ with $j < i$. Now, by fact 3, it is clear that $g_i$ has to change of group by processing \texttt{Group.List}.

Fact 5. Each term $g_i$ changes of group at most $t$ times.

It is a consequence of facts 2 and 3.

Fact 6. The function \texttt{Learn.Dlist} halts in time polynomial in $s$.

By facts 4 and 5, the \texttt{while} instruction of \texttt{Learn.Dlist} is processed at most $st$ times. Now, by fact 1 we get the result.

Now, let us suppose that $f$ is a log $n$-decision list. The following lemma says that, drawing examples under the universal distribution, with high probability a set $G$ including all terms of $f$ can be obtained in polynomial time.

Lemma 3.2 Let $f$ be a log $n$-decision list and let $f_i = (f_{mi}, fb_i)$ for $i = 1, \ldots, t$ be the terms of $f$. Let $c$ be a constant such that for all $i = 1, \ldots, t$, it holds $K(f_{mi}) < c \log n$. Let $\epsilon$ be a real number such that $0 < \epsilon < 1$. There exists an integer $n_c$ depending only on $c$ such that for all $n \geq n_c$, if we draw $n^{c^2}/\epsilon$ examples according to $m$, with probability greater than $1 - \epsilon$ a set $G$ including all terms of $f$ can be obtained in time $O(n^{2(c^2)/\epsilon^2})$.

Proof The idea is from [3] (see also [4]). If we draw $n^{c^2}/\epsilon$ examples according to $m$, it holds the following,

Fact. For all $n \geq n_c$, where $n_c$ is an integer depending only of $c$, with probability greater than $1 - \epsilon$ all examples of the following form will be drawn.

For each monomial $m$ over $n$ variables with $K(m) < c \log n$: the example vectors $0_m$, defined as the vectors that satisfy $m$ and have 0 entries for all variables not in $m$; the example vectors $1_m$, defined as the vectors that satisfy $m$ and have 1 entries for all variables not in $m$.

This fact is shown in the proof of theorem 3 of [3]. Now, we obtain the set $G$ with the following procedure:

Draw $n^{c^2}/\epsilon$ examples according to $m$. Let $E$ be this set of examples.

$G := \emptyset$

For each pair of examples in $E$ do

Let $m$ be the monomial which contains $x_i$ if both examples have '1' in position $i$, contains $\overline{x}_i$ if both examples have '0' in position $i$, and does
not contain variable $x_i$ otherwise ($1 \leq i \leq n$)

$G := G \cup \{(m,0),(m,1)\}$

endfor

Let us call this procedure Get_set. Using the fact above, it is clear that Get_set satisfies the lemma.

Finally, we are ready to show simple pac-learning for log n-decision lists. Let $f$ be as in lemma 3.2 and let $t$ be the time of Learn_Dlist running on a set that includes all terms of $f$. With probability at least $1 - \epsilon$, the algorithm

Get_set($G$)
Let $s \in O(n^{2(c+2)}/\epsilon^2)$ be the cardinality of $G$
Simulate Learn_Dlist for $t(s)$ steps

learns $f$ exactly. Removing the equivalence queries in the standard way (as in the proof of theorem 2.5, see [1] and [2]) we obtain a pac-learning algorithm, say $A$, with the following property,

$\Pr (m(g_A \neq f) < \epsilon) \geq$

$\Pr (m(g_A \neq f) < \epsilon | \{f_1, \ldots, f_l\} \subseteq G) \cdot \Pr (\{f_1, \ldots, f_l\} \subseteq G) >$

$(1 - \epsilon)(1 - \epsilon) \geq 1 - 2\epsilon$

where $g_A$ denotes the output list of $A$. Therefore, we have shown

**Theorem 3.3** Let $D_c$ be the class of log $n$-decision lists having all their terms Kolmogorov complexity bounded by $c \log n$. The $D_c$ classes are simple pac-learnable.

## 4 Conclusions

Li and Vitányi gave in [3] several examples of simple pac-learning. The outline of their proofs is the following. First a certain property $P$—that depends on the class $C$ they try to learn—is shown to happen with high probability drawing examples under $m$. Second, assuming that property $P$ happens, a pac-learning distribution-free algorithm—frequently an Occam algorithm—for $C$ is given. This work shows that, in some cases, a learning via equivalence queries algorithm may be easier to get than a pure pac-learning distribution-free or an Occam one.

We have some evidences of the fact that the similar learning scheme used in this paper, namely:

1. With high probability get property $P$.

can be improved in such a way that more general classes of decision lists would be simple pac-learnable. If we confirm these results, they will be written in a separate paper.
In a recent paper [6], an algorithm for exact learning, via equivalence queries, of decision lists has been independently obtained by Simon. Given a set of Boolean functions $F$, it can be defined, in a natural way, the class $\mathcal{D}_F$ of decision lists having all their terms functions in $F$. Assuming that $F$ is known, Simon's algorithm, denoted by $\text{Declist}$, learns $\mathcal{D}_F$ in time polynomial in the cardinality of $F$. Algorithms $\text{Declist}$ and $\text{Learn}_\text{Dlist}$ are similar. Note that if $F$ is known, the set $G$ including all terms of the target list needed by $\text{Learn}_\text{Dlist}$ can be obtained in linear time, just consider:

$$G = \bigcup_{f \in F} \{(f,0),(f,1)\}.$$ 

Acknowledgements

I discussed ideas of this work with José L. Balcázar who introduced me topics of learning theory. Ricard Gavaldà showed me the proof of theorem 2.5.

References


Recent Research Reports


LSI–94–49–R. “A posteriori knowledge: from ambiguous knowledge or undefined information to knowledge”, Matías Alvarado.


Copies of reports can be ordered from:

Nuria Sánchez
Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Pau Gargallo, 5
08028 Barcelona, Spain
secrelsi@lsi.upc.es