LEARNING MACHINE FOR PROBABILISTICALLY DESCRIBABLE CONCEPTS

Marek Karpinski*) and Zbigniew W. Ras**)

*) University of Bonn, Institute of Informatics, 5300 Bonn 1, West Germany

**) University of North Carolina, Dept. of Comp. Science, Charlotte, N.C. 28223, USA

ABSTRACT. A learning machine (see [9]) consists of a learning protocol together with a deduction procedure. Learning protocol specifies the manner in which information about the concepts is obtained from the outside. Deduction procedure is the mechanism by which a correct recognition algorithm for all concepts to be learned is deduced. There are m concepts to be learned. They are represented as terms in a probabilistic DNF form. The goal of this paper is to propose an efficient parallel algorithm on a PRAM (see [2]) for learning these concepts with a minimal error. Information about them comes from three types of queries: equivalence, membership and probabilistic queries. There is a treshold k describing the maximal number of conjuncts allowed in the final descriptions of concepts produced by a learning machine. It can happen that the descriptions known by the teacher contain more conjuncts per concept than k. The goal of the first step of our deduction procedure is to learn descriptions of concepts in the form known by the teacher. We apply here a generalization of a strategy proposed by Angluin (see [1]). In the second step, we apply the strategy which is similar to the one proposed by Ras and Zemankova (see [7]). This way we minimize the number of conjuncts in the descriptions of concepts by taking the advantage of a growing language. The error learning after these two steps is equal to zero. The last step is based on merging two closest conjuncts in one of the terms (in DNF) representing the current outcome of the deduction procedure. This step is repeated until there is no term in DNF (describing one of the concepts to be learned) which contains more conjuncts than the treshold k. The final output of the learning machine is a collection of optimized concepts descriptions in terms of a growing language.

INTRODUCTION. The main goal of a deduction procedure is to infer descriptions of concepts (decision rules) from a set of data (training events) gathered through an appropriate learning protocol. There have been many deduction procedures and learning protocols suggested. ID3 system proposed by Quinlan (see [6]) generates a decision tree for each concept to be learned. Nodes of his decision trees are labeled by attribute names and edges by corresponding attribute values. His

to be published in the Methodologies for Intelligent Systems, II, Z.W. Ras and L. Saitta (Editors), North Holland, 353-362

algorithm selects an attribute and partitions the training examples into disjoint subsets characterized by values of the attribute. The selection of attributes is guided by the expected entropy function assigned to each concept. The general idea is to choose an attribute which gives the maximal reduction of information uncertainty for a concept to be learned. AQ11 system proposed by Michalski (see [3,4]) generates a set of disjoint stars from a set of training events and selects from each star the best complex (conjunct) according to an optimality criterion. Angluin (see [1]) gives a polynomial algorithm to learn a precise description of a concept represented as a k-term DNF formula. The above methods do not say what to do if there are several objects assigned to one event and only some of them represent a concept to be learned. Such concepts are probabilistically describable. Ras and Zemankova (see [7,8]) assume that each training event is given with a probabilistic value and the language used by a learning machine has a probabilistic feature. They propose an interpretation of functors and concepts which does not guarantee the distributivity and idempotency laws. These laws are vital for the optimization process of decision rules. We suggest in this paper another interpretation (a probabilistic one) which preserves both idempotency and distributivity laws in the form of one-sided inequalities. The main problem stated in this paper is the following: there is a teacher who knows nonredundant descriptions of n concepts. These descriptions are in the form of k-term DNF formulas. There is a threshold given for the learning machine which gives the maximal number of conjuncts allowed in the final descriptions of concepts. The deduction procedure suggested by us has three main steps: a generalization of Angluin algorithm (see [1]), a generalization of Ras & Zemankova algorithm (see [7]) and finally a generalization algorithm based on the syntactic distance between terms. Our learning protocol is based on three types of queries: equivalence, membership and probabilistic queries. We assume that the learning machine knows the number of conjuncts in a nonredundant description of each concept.

BASIC DEFINITIONS.

In this section we introduce the notion of an event space (see [4,5]), a syntactic distance between events and sets of events, a formalized language L(S,C) which will be used to manipulate rules (terms describing concepts) in a knowledge base and finally we propose an interpretation of L(S,C) in the algebra of probabilistic functions.

Let us assume that X is a set of objects and $a_1, a_2, \ldots, a_n$ is a list of selected attributes used to describe them. By $Dom(a_i)$ we mean a finite set of values of the attribute a_i, $i \in I$. By an *event space E* we mean the cartesian product $Dom(a_1) \times Dom(a_2) \times \ldots \times Dom(a_n)$. An *information* *system* is a pair S = (E, D) , where E is a subset of ***E*** called the set of *observed events* and D is a function from E into the set of positive integers. Function D describes the frequency with which the examples of of each event from E occur in **X**. More precisely, the equation D(e) = k means that the number of objects in **X** described by an event e is equal to k. Following the definitions introduced in [2], we assume that attributes are either unordered, totally-ordered or structured. Furthermore, structured attributes are either ordered or unordered. They are ordered (unordered) if leaf values constitute an ordered (unordered) set.

Example 1. For simplicity reason the attributes are represented in LISP notation.
totally-ordered attribute
(integer 1 2 3 4 5 6 7 8 9)
unordered attribute
(color blue red green black orange yellow)
structured unordered attribute
(shape (polygon (3-sided triangle) (4-sided rectangle square trapezoid)) (oval circle ellipse))
structured ordered attribute
(some (couple 2) (few 2 3 4 5 6 7 8) (several 5 6 7 8 9 10))

By a *syntactic distance between values of an attribute* a_i we mean a function $\mathbf{d_a}$ from $Dom(a_i) \times Dom(a_i)$ into the interval <0,1>. The definition depends on the type of an attribute a_i. It is given below

If a_i is an unordered attribute and v_1 , v_2 are distinct elements in $Dom(a_i)$, then $\mathbf{d_a}(v_1,v_2) = 1$. If $v_1 = v_2$, then $\mathbf{d_a}(v_1,v_2) = 0$.
If a_i is a totally-ordered attribute and $(v_1, v_2, v_3,..., v_k)$ is an ordered sequence of all its values, then $\mathbf{d_a}(v_i,v_j) = |j-i|/(k-1)$
If a_i is an unordered structured attribute and v_1 , v_2 are its different values, then $\mathbf{d_a}(v_1,v_2)$ is defined as $1/(2^{**}depth(\mathbf{c}(v_1,v_2)))$, where $\mathbf{c}(v_1,v_2)$ is the nearest common ancestor for v_1 and v_2. The depth of the root of the tree is zero. If $v_1 = v_2$, then $\mathbf{d_a}(v_1,v_2) = 0$.
The distance between values of an ordered structured attribute can be defined either as the distance between values of an ordered attribute or as a combination of the distance between values of an ordered attribute and the distance between values of an unordered structured attribute.

Let $e_1 = (v_1,v_2,v_3,...,v_n)$, $e_2 = (w_1,w_2,w_3,...,w_n)$ be any two events. By a *syntactic distance between events* we mean a function d from $E \times E$

into the set of positive real numbers defined as follows

$$d(e_1,e_2) = \sum_{i=1}^{n} d_a(v_i,w_i) .$$

By a *syntactic distance between sets* E_1, E_2 *of events* we mean a function **d** from $2^E \times 2^E$ into the set of positive real numbers defined as follows $d(E_1,E_2) = \min\{ d(e_1,e_2) \mid e_1 \in E_1 , e_2 \in E_2\}$.

Let E be the set of observed events. By a *probabilistic function* on E we mean any total function from E into the interval <0,1>. Let F_E be the set of all probablistic functions on E. The *algebra* $A(F_E) = (F_E,\#,\&,-)$ of probabilistic functions on E is defined below

For any f_1 , f_2 in F_E

$(f_1 \# f_2)(e) = f_1(e) + f_2(e) - f_1(e)*f_2(e)$

$(f_1 \& f_2)(e) = f_1(e) * f_2(e)$

$(-f_1)(e) = 1 - f_1(e)$

Fact 1. Let f_1, f_2, f_3, f are elements of F_E and $g = f - f*f$. Then

1) $f \& f \leq f$, $f \leq f \# f$
2) $f_1 \& f_2 = f_2 \& f_1$, $f_1 \# f_2 = f_2 \# f_1$
3) $(f_1 \& f_2) \& f_3 = f_1 \& (f_2 \& f_3)$, $(f_1 \# f_2) \# f_3 = f_1 \# (f_2 \# f_3)$
4) $-(f_1 \# f_2) = (-f_1) \& (-f_2)$, $-(f_1 \& f_2) = (-f_1) \# (-f_2)$
5) $(-f) \# f = 1 - g$, $(-f) \& f = g$
6) $(f \& f_1) \# f = f + f_1*g$, $(f \# f_1) \& f = f - g*(1-f_1)$
7) $(f \& f_1) \# (f \& f_2) = f \& (f_1 \# f_2) + f_1*f_2*g$
8) $(f \# f_1) \& (f \# f_2) = f \# (f_1 \& f_2) - g*(1- (f_1 \# f_2))$

We say that a probabilistic function f on E is *crisp* if f takes only values from the set {0,1}. Let FC_E be the set of all crisp probabilistic functions on E. The sub-algebra $A(FC_E) = (FC_E,\#,\&,-)$ of the algebra $A(F_E)$ is a Boolean algebra. This observation follows automatically from Fact 1 and from the fact that crisp functions satisfy the axiom $f = f*f$.

Let's define the *language* L(S,C) *of probabilistic DNF formulas* . We assume that S=(E,D) is an information system and C is a set of concepts to be learned.

The set of *atoms* is a least set such that

1) A pair $([V_1,V_2,V_3,\ldots,V_n], w)$ is an atom. We assume that $w \in (0,1]$ and either $V_i = Dom(a_i)$ or $V_i \in Dom(a_i)$ for any $1 \leq i \leq n$. If $V_i = Dom(a_i)$, then V_i is replaced by the symbol "*". If w = 1, then $([V_1,V_2,V_3,\ldots,V_n],w)$

is replaced by $[V_1,V_2,V_3,\dots,V_n]$.
2) A pair (c, w) is an atom. We assume that $c \in C$ and $w \in (0,1]$. If $w = 1$, then (c, w) is replaced by c.

The set of *terms* is a least set such that
1) all atoms are terms
2) if t_1, t_2 are terms then (t_1+t_2) is a term.

The set of *formulas* is a least set such that
1) if t_1, t_2 are terms then $(t_1=t_2)$, $(t_1\leq t_2)$ are formulas
2) if $\mathbb{a}$, $\mathbb{b}$ are formulas then $(\mathbb{a} \vee \mathbb{b})$, $(\mathbb{a} \wedge \mathbb{b})$, $-\mathbb{a}$ are formulas.

We say that (t_1+t_2) is a disjunction of two terms t_1 and t_2. By k-term in DNF we mean any disjunction of k atoms.

Let us define the interpretation J_S of the language L(S,C) in the algebra $A(F_E) = (F_E,\#,\&,-)$ and two-valued Boolean algebra.
1) $J_S(([V_1,V_2,V_3,\dots,V_n], w)) = f$, where $f \in F_E$ and f is defined as follows
$f(e) =$ (if $e \in V_1 x V_2 x V_3 x \dots x V_n$ then w, else 0)
2) $J_S((c,w)) = g$, where $g \in F_E$ and g is defined as follows
$g(e) = w^* f(e)$, where $f = J_S(c)$ and $f \in F_E$.
3) $J_S(+) = \#$, $J_S(=) =$ (equality for functions), $J_S(\leq) =$ (one sided inequality for functions)

Atoms of L(S,C) correspond to conjuncts. This is why we have only one functor "+" in L(S,C). The reason of introducing the algebra $A(F_E)$ was to show another possible interpretation of terms and formulas (built from concepts and attribute values) on the boundary region (see [7,8]). The axioms for manipulating terms are strictly dependent on this interpretation. The interpretation proposed in [8] does not guarantee the idempotency and distributivity laws. Our interpretation guarantee both laws in the form of one-sided inequalities. The lack of these laws forced Ras and Zemankova (see [7]) to use a growing language in order to reduce the number of conjuncts in terms (in DNF) describing concepts and at the same time to keep the semantical meening of all terms the same. It has to be realized that our interpretation of formulas does not improve this optimization problem very much. However it can be said that the axiom

$$(f \,\&\, f_1)\ \#\ (f \,\&\, f_2) \geq f \,\&\, (f_1\ \#\ f_2)$$

corresponds to the specification step.
The strategy for learning the optimal DNF's proposed by Michalski in [3,4,5] is using very often the distributivity laws. In our case they do not hold. This justifies why we want to adopt and slightly generalize the

learning strategy proposed by Angluin (see [1]) in order to get the starting terms in DNF describing m concepts to be learned by our learning machine. Simply her strategy avoids the distributivity laws.

DEDUCTION PROCEDURE

In this section we present an algorithm for learning m concepts in parallel. The number of processors is equal to the number of attributes used to describe concepts. Each processor p_i, $1 \leq i \leq n$ has access to three oracles $Oracle1_i()$, $Oracle2_i()$ and $Oracle3_i()$. The algorithm has three main procedures. The first one is a generalization of Angluin algorithm (see [1]). The second is a modification of Ras and Zemankova algorithm (see [4]) and the last procedure is a generalization algorithm based on the syntactic distance between terms.
Initially, we present our algorithm for two-valued attributes. The interpretation J_S will be in the algebra $A(FC_E) = (FC_E,\#,\&,-)$ of crisp probabilistic functions on E. In this case, a processor p_i, $1 \leq i \leq n$ needs an access only to $Oracle1_i()$ and $Oracle2_i()$. Values of an attribute a_i will be denoted by v_i, v_i'. We call them literals. We will write v_i instead of $(v_i')'$.

Two terms t_1, t_2 are *equivalent* if the statement $t_1 = t_2$ is true in the interpretation J_S. Assume that t is a term in DNF and $J_S(t)(e)=1$ where $e = [w_1, w_2, \ldots, w_i, \ldots, w_n]$. We say that w_i is *sensitive* in e with respect to $J_S(t)$ if $J_S(t)(e')=0$ where $e'=[w_1, w_2, \ldots, w_i', \ldots w_n]$. The set of all w_i sensitive in e with respect to $J_S(t)$ is denoted by $S_t(e)$. Assume that $t = t_1+t_2+\ldots+t_k$ is a k-term in DNF. We say that t is *non-redundant* if a term received by droping in t any atom t_i is not equivalent to t. It has been proved in [1] that if $t = t_1+t_2+\ldots+t_k$ is non-redundant then there is a sequence of events $(e_1, e_2, \ldots, e_k)$ such that for any i, $J_S(t_i)(e_i)=1$ and $J_S(t_j)(e_i)=0$ for $j \neq i$. Let's call the event e_i a *generator* for t_i for $i \in I$. By L_i we mean a set of literals not listed in t_i which has a non-empty intersection with a set of literals listed in t_j for any j where $j \neq i$. By R_i we mean a set of literals listed in e_i which satisfy the following property: if a literal is in R_i then there is t_j $(j \neq i)$ in which this literal is not used. Both sets L_i and R_i can be obtained easily from the set of generators. The set L_i is called a discriminant set and it is used to find a generator for t_i.

The deduction procedure $LEARN_1()$. We assume that the value k and the values of attributes $a_1, a_2, \ldots, a_n$ are known to the deduction procedure. The deduction procedure has to learn a nonredundant k-term t' in DNF equivalent to t. There are n processors $p_1, p_2, \ldots, p_n$. Each processor p_i has access to two oracles $Oracle1_i()$, $Oracle2_i()$. $Oracle1_i()$ outputs 1 on an event e if $J_S(t)(e)=1$. Otherwise it outputs 0. $Oracle2_i()$ outputs 1 on a term t' if t' is equivalent to t. Otherwise it returns an event which is a counterexample for the equation $J_S(t) = J_S(t')$.

Let's start the description of a deduction procedure under the assumption that each processor p_i knows both sets L_i and R_i. One of the processors has to ask the second oracle if t' is a null term. Assume that the answer is "no" and the counterexample e is read by all processors. Each processor p_i will replace in e all values listed in L_i by their dual values (v_j is replaced by v_j' and v_j' by v_j) and then it will ask $Oracle1_i()$ if $J_S(t)$ has value 1 on the new obtained event. If the answer is "yes", then the new obtained event is a generator for t_i. It can be proved that minimum one processor (let's say p_j) will get a positive answer from the Oracle1(). It this case, the generator e_j for t_j is sent by p_j to all other processors through the global PRAM memory. Each processor p_i will replace the i-coordinate of e_j by the dual i-coordinate and will ask $Oracle1_i()$ if $J_S(t)$ has value 1 on the new event. If answer is 1, then the i-coordinate of e_j is not sensitive. If answer is 0, then the processor p_i sends the message to the processor p_j that the i-coordinate of e_j is sensitive. Processor p_j has to add all sensitive coordinates of e_j to the set R_j. The obtained set gives all the literals which have to be listed in the new generated term t_j'. Processor p_j will ask the $Oracle2_j()$ if $J_S(t) = J_S(t')$. $Oracle2_j()$ either outputs "yes" or gives the next counterexample. This counterexample is used to find a generator of the next t_i, and so on.

The procedure requires that each processor p_i knows sets L_i and R_i. This assumption in Angluin paper (see [1]) is not necessary since the number of possible options for these sets is bounded by a polynomial function with respect to n. Her procedure tries all these options in order to find the correct one.

The assumption that attributes have only two values is not necessary. Clearly, we can not replace a many-valued attribute by a set of two-valued attributes because the number of conjuncts in DNF will increase considerably (we want to minimize this number). The generalization of Angluin's algorithm to many-valued attributes can be done easily. Assume that $Dom(a_i)$ is a minimum 3-element set, where a_i is an attribute. Two cases have to be considered:

1) $Dom(a_i)$ is a subset of L_i,

2) there is v_0 $(1 \leq i \leq k)$ in $Dom(a_i)$ such that $v_0 \epsilon L_i$.

In the first case there are two values v_1, v_2 in $Dom(a_i) \cap L_i$ listed in two conjuncts t_1, t_2 respectively. We can choose v_1, v_2 in a such a way that the corresponding conjuncts will differentiate on minimum two attributes. To eliminate these conjuncts we have to swop v_1 and v_2. Let us assume that t_3 is a conjunct in which v_3 is listed, where $v_3 \epsilon Dom(a_i)$. Either t_3, t_1 or t_3, t_2 will differentiate on minimum two attributes . If t_3, t_1 differentiate, then t_3 is replaced by t_1. If t_3, t_2 differentiate, then t_3 is replaced by t_2. This case requires to add one more loop in Angluin's algorithm in order to find a proper replacement for all elements in L_i.

In the second case all elements from $L_i \cap Dom(a_i)$ are replaced by v_0.

Let us assume that J_S is an interpretation in the algebra $A(\mathbf{F}_E)$ of probabilistic functions on E. We need one more definition to describe the procedure *LEARN*$_1$*()*. We say a term t is weakly equivalent to a term t' if the following two conditions hold:

1) $J_S(t)(e) \geq 0$ iff $J_S(t')(e) \geq 0$ for any event e,

2) $J_S(t)(e) = 1$ iff $J_S(t')(e) = 1$ for any event e.

Three oracles are needed to extend Angluin algorithm to probabilistic case in a natural way.

Oracle1$_i$() will output 1 on an event e if $J_S(t)(e) \geq 0$,

Oracle2$_i$() will output 1 on a term t' if t' is weakly equivalent to t.

Oracle3$_i$() will output a probabilistic value p on a term t' (p is the probability that events described by t' are described by t).

Initially we use only Oracle1$_i$() and Oracle2$_i$(). Oracle3$_i$() is needed at the end of *LEARN*$_1$*()* in order to assign probabilistic values to conjuncts learned by using the previous two oracles.

The procedure *LEARN*$_1$*()* will output a k-term t' in DNF equivalent to the term t. It may happen that the number k is larger than the threshold set up for the learning machine. The threshold gives the maximal number of atoms (conjuncts) allowed in the final form of t'. We will take the adwentage of a growing language to shrink the term t' if k is larger than the threshold.

Let us assume that $s_1', s_2', \ldots, s_m'$ is a sequence of terms in $L(S,\emptyset)$ describing concepts $c_1, c_2, \ldots, c_m$ respectively. This fact is expressed in $L(S,\{c_1,c_2,\ldots,c_m\})$ as a sequence of formulas $c_i = s_i'$ $(1 \leq i \leq m)$ true in the interpretation J_S. Each s_i' is nonredundant. The deduction procedure *LEARN$_1$()* learns a sequence of terms $s_1, s_2, \ldots, s_m$ equivalent to s_1', $s_2', \ldots, s_m'$ respectively. The sequence of formulas $c_i = s_i$ $(1 \leq i \leq m)$ is passed as an input to the deduction procedure *LEARN$_2$()*. *LEARN$_2$()* outputs a new sequence of formulas $c_i = r_i$ $(1 \leq i \leq m)$ in terms of a growing language. These new formulas are true in the interpretation J_S. Saying more precisely *LEARN$_2$()* has to decide in which order the concepts $c_1, c_2, \ldots, c_i$ $(i \leq m)$ has to be taught in order to use them in a most efficient way in the process teaching the concept c_{i+1}. Procedure *LEARN$_2$()* denotes the algorithm proposed by Ras and Zemankova in [4].

The sequence of formulas $c_i = s_i$ $(1 \leq i \leq m)$ is passed as an input to the deduction procedure *LEARN$_3$()*. It may happen that the number of atoms (conjuncts) in some s_i $(1 \leq i \leq m)$ is still higher than the threshold set up for the learning machine. *LEARN$_3$()*, which is a generalization procedure will merge the nearest (with respect to the syntactic distance) atoms (conjuncts) in s_i and will check if the threshold is higher than the number of atoms in the modified version of s_i. If so, then *LEARN$_3$()* is called again. The idea of merging the nearest atoms of the input term s_i is justified by the need of adding to s_i a minimal number of new events (not covered by s_i) to produce a generalized version of s_i.

BIBLIOGRAPHY.

[1] Angluin, D., "*Learning k-terms DNF formulas using queries and counterexamples* ", Yale University, August 1987

[2] Karp, R.M., Ramachandran, V.L., "*A survey of parallel algorithms for shared-memory machines*" , University of California at Berkeley, 1988

[3] Michalski, R.S., Larson, J.B.,"*Selection of most representative training examples and incremental generation of VL1 hypothesis - the underlying methodology and description of programs ESEL and AQ11* ", Report No. 867, Dept. of Comp. Sci., Univ. of Illinois, Urbana, 1978

[4] Michalski, R.S., "*Synthesis of optimal and quasi-optimal variable-valued logic formulas* ", Proceedings of the MVL'75 in Bloomington, Indiana, 76-87

[5] Michalski, R.S., Stepp, R.E., "*Learning from observation: conceptual clustering* ", Chapter 11, Machine Learning - Artificial Intelligence Approach, Tioga Publishing Company, Palo Alto, R.S. Michalski, J.

Carbonell, T. Mitchell (Editors), 1983

[6] Quinlan, J.R., "*Learning efficient classification procedures and their application to chess end games* ", Chapter 15, Machine Learning - Artificial Intelligence Approach, Tioga Publishing Company, Palo Alto, R.S. Michalski, J. Carbonell, T. Mitchell (Editors), 1983

[7] Ras, Z.W., Zemankova, M., "*Learning driven by the concepts structure*", Proceedings of IPMU'88 at Urbino, Italy, Lecture Notes in Computer Science, Springer Verlag

[8] Ras, Z.W., Zemankova, M., "*Learning in knowledge based systems, a possibilistic approach* ", Proceedings of MFCS'86 in Bratislava, Czechoslovakia, Lecture Notes in Computer Science, Springer Verlag., No. 233, 630-638

[9] Valiant, L.G.,"*A theory of the learnable* ", Communications of the ACM, November 1984, Vol. 27, No. 11, 1134-1142

[10] Valiant, L.G., "*Deductive learning* ", Phil. Trans. R. Soc. Lond. A 312, 1984, 441-446