LEARNING IN THE LIMIT IN A GROWING LANGUAGE

Ranan B. Banerji
Department of Mathematics and Computer Science
Saint Joseph's University
Philadelphia, PA 19131
(rbanerjii@sju.edu)

ABSTRACT

This paper describes the design of an algorithm which learns a class of theories in the limit in a sublanguage of a first order language. The languages chosen are richer than those handled previously - needing a more efficient search through the version space. The language is not initially known to the system, but is learned together with the theory.

1 INTRODUCTION AND BACKGROUND

The word "learning" will be used here to mean "developing class descriptions from examples of the members of the class". The work described here is a continuation of a series of efforts [1,2,3,4][to simulate the kind of learning where the efficiency of the learned description increases with increase in knowledge. This needs the ability to use the name of one class in the description of another. Such a process is often called "Multilevel Learning", since the wherewithal for the description grows from the initial features of the language to include the names of previously learned concepts as well as names of internally generated concepts used as additional features. The concepts as well as the features can be relations as well as classes. The descriptions of relations can be learned using the same technique as is used in learning the descriptions of classes.

The formal basis for this work is the concept of "Learning in the Limit" (developed by Gold [6] as a model for learning languages and extended by Shapiro [5] to the learning of theories from facts in first order logic). Since the descriptions of classes can be looked upon as sets of Horn Clauses, Shapiro's general technique is applicable to the learning of the descriptions of classes and relations. The technique does not need that the examples be presented to the learning mechanism by a helpful teacher, as was assumed by some of the authors mentioned here.

In this present work wo have used a more general language than those usod in Shapiro's experiments. The language chosen here seems more suitable to the purpose of concept learning. Certain improvements to the technique have been made for pruning the search for descriptions in the chosen language. Also, unlike Shapiro's work, it has not been assumed that the language is available to the learning algorithm: the algorithm learns the language as it learns the descriptions. Apart from this, the work follows Shapiro's guidelines quite faithfully.

In what follows we shall assume acquaintance with Shapiro's approach and describe our language and algorithm in brief.

II THE CLASS OF LANGUAGES

In the first order languages used here for forming descriptions, there are variables, constants and unary function symbols. No binary or higher arity function symbols are used. An atomic sentence or atom consists of a term followed by an equals sign followed by a constant. A term is either a variable or a function symbol followed by a term enclosed in parentheses. A predicate is either an atom or consists of a predicate letter of arity n, followed by n terms separated by commas. Such a predicate is called a defined predicate. If all the terms occuring in a defined predicate are variables, then the predicate is called a leftside. A sentence consists of a Horn clause, whose head is a leftside and whose body contains predicates.

For the purposes of this work, some of the predicate letters are considered to be in a special class: these are called concept letters. A leftside formed with a concept letter is called a definlend.

Predicates formed by concept letters are the ones which name concepts to be learned. Predicate letters other than concept letters can be generated internally by the learning algorithm to further simplify complex descriptions. These are what people often call "features" in pattern recognition parlance. The processes of "feature extraction" and compression will be described in the section on algorithms.

Sentences as described above form the hypothesis language L of Shapiro. The observation language L consists of ground sentences, which are defined to be those whose head is a definiend and whose body consists entirely of atoms.

A theory will be defined to be a set of sentences and the definition of proofs and the definition of a theory implying a sentence is

standard. Instead of defining the model of a theory by an abstract mathematical structure, we shall define the word syntactically. Given a theory, a ground sentence P:-B will be called minimal in the theory if it is implied by the theory and no ground sentence P:-A,withA a subset of B, is implied by the theory. The model of the theory is the set of all ground sentences minimal in the theory.

It can be shown that given a ground sentence one can predict how long a proof of it from a theory is going to be. This predictability is required by Shapiro's approach to learning. Formal proof of this predictability and other claims will be published elsewhere [7].

For a proper application of Shapiro's approach, we need proofs to be in a form in which at every step one of the resolvents is a ground sentence. An algorithm has been developed by us which, given any proof, can convert it into one of this desired form. Such a proof can be used to initiate a process called "contradiction backtrace" for what is known as "credit assignment" in the field, i.e. to find out which sentences in a theory lead to proofs of false sentences. If a fact defines a certain ground sentence to be false and a proof for it exists from a theory, then the axiom to blame is found as follows. At each step of the proof the algorithm asks of the environment whether the ground resolvent is true (the sentence being in L , it is legal to ask the question). If it is false, one of the axioms used in its proof must be false. Else the axioms involved in the proof of the other, non-ground resolvent must be false. The algorithm asks the same question at the different steps of the "guilty" tree till it isolates the false axiom.

The learning algorithm described in the next section will use this algorithm as well as the check on provability described above.

1Il THE ALGORITHM AND ITS CONVERGENCE

At any point of its operation, the learning program's knowledge has two components: the knowledge of the language and the theory developed so far. The structure of the theory has been discussed already. The knowledge of the language associates with each concept letter a set of atoms which occur in some positive examples of the concept (i.e. facts with the concept at the head and marked true).

There are several major procedures used by thelearning algorithm. The first one is:

(I)ADD: This procedure is invoked when the program encounters a positive example of a concept not implied by the current theory. It adds to the language by associating with the predicate at the head of the fact all the atoms in the body of the fact (unless the atom was already there). ADD also modifies the theory. At the first occurence of a positive example of the concept, the theory is augmented by P:-, where P is the definiend corresponding to the concept. The body of the sentence is empty, indicating that "everything is a P." Else it augments the theory by adding the sentences P:-A, where A is one of the atoms just added to the language. There is one such sentence added for each new atom.

(The reader will note that the theory learned by the algorithm has a lot of disjunctions, but the disjuncts are not the positive examples themselves, as in rote learning. Rather the disjuncts are alternative generalizations of the examples.)

The next major procedure is

(2) SHRINK: Invoked when it is found that the theory Implies a ground sentence which has been marked false. First the algorithm locates the (or one of the) sentences at fault in the theory (asking questions of the environment as described in the previous section). If the culprit is a ground sentence, then SHRINK adds to (i.e. forms the conjunction with) the body one of the atoms in the language associated with the predicate at its head. In doing so care is taken that the resulting sentence is not implied by any other sentence of the theory.

It can be shown that if the model is consistent (i.e. if no false ground sentence has a body which contains the body of a true ground sentence with the same head), then the algorithm given below converges to a correct theory for any finite model. However, the resulting theory can be unnecessarily complex. Moreover, since these processes only learn theories with ground sentences, no infinite model can be explained with such theories. This drawback can be rectified with the following procedure.

(3) GENERALIZE: This process is invoked onlv after the two previous processes working in two cooperating loops have resulted in a theory with ground sentences explaining all the known facts. GENERALIZE is brought into action if the previous loops have added to or shrunk the ground part of the theory in their current invocation. At this point, the algorithm looks for all pairs of sentences in the theory in which the body of the first is subsumed by an instance of the body of the second. In this case, the involved part of the body of the first is replaced by the substitution instance of the head of the second.

One other process is invoked for compression of the sentences of the theory. It is called DREAM and is merely a generalization of an application of the distributive law of Boolean logic. DREAM is responsible for introducing the internally generated feature names. It can be shown that DREAM does not change the set of sentences implied by a theory.

A skeleton of the algorithm is shown in Figure 1.

The algorithm works infinitely, verifying the current theory continuously, changing it only when facts demand it. The point to be made here is that the algorithm is such that as long as a theory exists in the language, it can be found in a finite period of time and no modification is needed after that.

FIGURE 1

```
Begin
  F+:=Φ;F_:=Φ;L:=Φ;T:=Φ;
  Do forever
    begin
      read a fact <M,v> and place M in Fv;
      FLAG:=false;
      while there is a M∈F+ not implied by T or
            there is a M∈F_ implied by T do
        begin
          while there is an M∈F_ implied by T do
            begin
              locate Q:-B∈T responsible for error;
              if B contains a defined predicate
                then T = T - {Q:-B}
                else begin T:=SHRINK(T,L,Q:-B);
                           flag:=true end
            end
          if there is P:-A∈F+ not implied by T
            then (T,L):=ADD(T,L,P:-A)
        end;
      if FLAG then for every pair of sentences in
        T allowing GENERALIZE, apply it, provided
        no M∈F_ is implied by the sentence added;
      apply DREAM to all applicable cases
    end
  end
end.
```

The main loop of the algorithm consists of two other loops, the first working with ADD and SHRINK and the other working with GENERALIZE and DREAM. It has already been indicated that the first loop always terminates with a ground theory for the finite set of facts currently available in F_+ and F_-. The second loop only introduces those generalizations that do not imply any known false facts — it does not remove any true known facts. It is possible that the generalizations introduced will explain true facts as well as false facts as the outer loop continues. It can, however be shown that in the kind of infinite models that theories of this class of lnaguages can express, sentences are bound to occur which lead to generalizations which explain only the true facts, provided that if any sentence of the theory contains two defined predicates in its body, the two predicates model disjoint sets of ground sentences. Thus the algorithm does learn theories of infinite models in the limit in this restricted class of theories. Finite models, of course are learned as soon as all the minimal true sentences are constructed by ADD and SHRINK.

IV DISCUSSION OF EFFICIENCY

There are two distinct phenomena that affect the time efficiency of the algorithm. One, there is the limit to the number of times the three successive inner loops to the program would be entered to make the theory compatible with the currently available examples. Calculation of the worst case complexity, as far as the inner loops are concerned, leads to the conjecture that the time complexity would be exponential in the number of atoms in the language conjecture. So the overall efficiency of the program really depends on whether it can develop the theory as soon as it has gathered the minimal number of atoms the theory needs.

The answer to this latter question depends very heavily on the second aspect of the efficiency, to wit, how many times the main loop will have to be entered before the correct theory can be discovered by the algorithm. Unfortunately, the answer to the question is entirely dependent on the property of the input sequence. It is our conjecture looking at some example theories that for each theory a minimal set of examples exist which suffices to make the algorithm discover the theory. One can only make a probabilistic estimate of how late it will be before the crucial examples are presented to the algorithm.

Presently an experiment is being carried out on this matter using a Markov model as source of examples. It is intended that this model will also be used in the theoretical estimation of the average efficiency of the overall algorithm.

V ACKNOWLEDGEMENTS

The work described in this paper was supported by the National Science Foundation under grant DCR-8504223 to the Saint Joseph's University. The preparation of the manuscript was supported by the Saint Joseph's University as a part of thecontribution to research in Machine Learning undertaken in cooperation with the Benjamin Franklin Partnership with the state of Pennsylvania and with Expert Systems International, Inc.

VI REFERENCES

1 Pennypacker, J. "An Elementary Information Processing Program for Object Recognition." Report No. SRC 30-1-63, Case Institute of Technology, Cleveland, OH, 1963.

2 Cohen, B., "Confucius: A Structural Pattern Recognition and Learning System." In Pror. the International Conference on Cybernetics and Society, Koyoto, Japan, 1978, p. 1443.

3 Sammut, C., "Learning Concepts by Performing Experiments." Ph.D. Dissertation, University of New South Wales, Kingston, 1981.

4 Banerji, R.B., Theory of Problem Solving: An Approach to Artificial Intelligence, American Elsevier, NY, 1969.

5 Shapiro, E., "Inductive Inference of Theories from Facts." Technical Report No. 193, Computer Science Department, Yale University, New Haven, CT, 1981.

6 Gold, E.M., "Language Identification in the Limit." Information and Control, 10, 447, 1965.

7 Banerji, R.B., "An Algorithm for Learning Theories in a Growing Language." Report from the Laboratory for Research in Computer Learning, Saint Joseph's University, Philadelphia, PA, 1987.