Learning Disjunctive Concepts with Distributed Genetic Algorithms

F. Neri      F. Zini
Dipartimento di Informatica, Università di Torino
Corso Svizzera 185, 10149 Torino (Italy)
{neri, flo}@di.unito.it

1. Introduction

Inducing concept descriptions from examples is a machine learning task that can be formulated as a search problem. Recently, genetic algorithms proved to be an appealing alternative for searching concept description spaces, because of their great exploration power and their suitability to exploit massive parallelism. [De Jong & Spears, 1991; Vafaie & De Jong, 1991; Janikov, 1992; Giordana & Sale, 1992; Giordana & Saitta, 1993].

A description of a concept, \( h \), is a rule

\[ \varphi \rightarrow h, \]  

where \( \varphi \) is a formula of a given language \( L \) and \( \rightarrow \) denotes implication. Given a rule like (1) and an example \( \xi \), the rule “covers” the example iff \( \varphi \) is true of \( \xi \). Then, the task of learning concept descriptions can be formulated as a set covering problem: given two sets, \( E(h) \) and \( C(h) \) of positive and negative examples of \( h \), respectively, we must find a rule that covers the whole set \( E(h) \) and does not cover any element of \( C(h) \).

In order to exploit genetic algorithms for searching the space of formulas (the elements of the population), we must reformulate the set covering problem by suitably defining the representation scheme for the formulas, the genetic operators and the fitness function.

The system described in this paper, REGAL [Giordana & Saitta, 1993], encodes first order logic (FOL) formulas \( \varphi \) into bit strings of fixed length. In order to make this possible, REGAL's representation language, \( L \), has been constrained to be finite, by defining a language template, \( R \), which represents the maximally complex formula in \( L \). The resulting language contains FOL formulas built up with conjunction, disjunction, negation, internal disjunction and existential and universal quantification.

In REGAL five genetic operators have been defined: the two-point and the uniform crossovers [De Jong, 1975; Goldberg, 1989], the generalizing and specializing crossovers, specifically designed for the task at hand [Giordana & Sale, 1992], and the seeding operator, primarily used to initialize the population, in order to start with a set of formulas, each covering at least one example of the concept \( h \) [Giordana & Saitta, 1993].

In the basic genetic algorithm an improved version of the sharing functions mechanism [Goldberg & Richardson, 1987] has been introduced, in order to allow the formation of subpopulations. This is useful when concepts have a multimodal structure, as will be discussed in the following sections.

2. Learning Disjunctive Concepts

The fundamental novelty of REGAL is its ability to learn disjunctive concepts. A disjunctive concept requires a description of the form:

\[ \varphi_1 \lor \varphi_2 \lor \ldots \lor \varphi_m \rightarrow h, \]  

where each formula \( \varphi_j \) \((1 \leq j \leq m)\) covers a region of the instance space and corresponds to a different mode of the concept. If the fitness function \( f(\varphi) \), associated to a formula \( \varphi \), takes into account, as usual, completeness and consistency [Michalski, 1983], then \( f(\varphi) \) is likely to exhibit several maxima, in correspondence to the modes of the concept.

† In presence of noise these strict conditions can be relaxed, by allowing some degree of inconsistency of the descriptions to be tolerated.
Learning disjunctive concepts is a difficult task, especially when some of the disjuncts are “small”, i.e., they cover a much smaller number of examples than others [Holte et al., 1989; Danyluk & Provost, 1993]. One way to deal with this problem is to learn a disjunct at a time [Michalski, 1983; Quinlan, 1990]. Learning one disjunct at a time is probably the most easy way to cope with the deceptiveness of the problem. By exploiting the theory of the niches and species [Deb & Goldberg, 1987], an alternative approach can be used, i.e., to learn all the disjuncts at the same time by encouraging the formation of subpopulations.

In previous papers both strategies have been tested, exploiting the mechanisms of crowding and sharing functions [Giordana & Sale, 1992; Giordana & Saitta, 1993]. Learning all the disjuncts at the same time seems more appealing, both for computational reasons and because, in principle, a globally better solution could be found. In this paper we propose a new approach to the problem, using a new type of sharing functions, which allow the order of magnitude of the selection operator complexity to be reduced with respect to existing approaches.

The new approach is outlined in the following. Let $\text{COV}(q)$ be the set of instances, belonging to the positive training set $E$, covered by formula $q$. We search for a description like (2) such that:

$$\bigcup_{j=1}^{m} \text{COV}(q_j) \supseteq E(h) \quad \text{and} \quad \bigcup_{j=1}^{m} \text{COV}(q_j) \cap C(h) = \emptyset$$

(3)

The new approach proposes to change the selection operator that chooses the elements to be mated in the population. A metaphor can be used to explain its basic principle: each example $\xi$ considers a formula $q$ covering it as one of its potential representatives in a parliament. The selection shall be done in such a way that each example in $E$ have the chance of being represented. More formally, let $R(\xi) = \{ q_i | q_i \in A(t), 1 \leq i \leq r \}$ be the set of formulas covering $\xi$, belonging to the population $A(t)$. Each $R(\xi)$ can be considered as a roulette wheel, associated to $\xi$, divided into $r$ sectors, each corresponding to a $q_i$, with a surface proportional to $p_i$:

$$p_i = p(q_i) = \sum_{q_k \in R(\xi)} f(q_k)$$

(4)

Let now $N = |E|, M = |A(t)|$ and $n \leq M$ be the number of formulas we want to select for reproduction. The selection process proceeds as follows. A number $n$ of examples are extracted with replacement from $E$ with equal probability each time. For each $\xi_j$ ($1 \leq j \leq n$), a corresponding roulette wheel is spinned and the formula corresponding to the sector where the ball halts is selected for reproduction. Notice that the same $\xi_j$ can be extracted more than one time and that a formula can be associated to more than one roulette.

For computing (4), the following fitness function is used:

$$f(q) = (1 + A \ z) e^{-\alpha w^\beta} + D$$

(5)

In (5), $w$ and $z$ are measures of $q$’s consistency and simplicity, respectively, whereas $0 \leq \alpha, \beta, A \leq 1$ and $D < 1$ are user-defined parameters. Notice that $f(q)$ is not required to depend upon $q$’s completeness, because the coverage of the formulas is taken into account in the very procedure of selection, introducing the sets $R(\xi)$’s.

Given a formula $q$ belonging to $A(t)$, let $s = |\text{COV}(q)|$. Then, $q$ occurs in $s$ different roulettes $R(\xi_k)$ ($1 \leq k \leq s$) with possibly different probability. The extraction of the examples from $E$ follows a multinomial probability distribution over $E$, in which each $\xi_j$ ($1 \leq j \leq N$) has a probability $1/N$ of being extracted. However, in order to simplify the problem, we will approximate the true distribution function with a binomial one, by simply considering the set $E$ partitioned into the subset $S$, of cardinality $s$, containing the examples $\xi_k$ ($1 \leq k \leq s$) corresponding to the $R(\xi_k)$’s, and in the complement set. Clearly, for each example extracted from $S$, $q$ will participate in the spinning of a roulette wheel. Then:

$$P_{\text{sel}}(q) = P(q \text{ is selected for reproduction}) = f(q) \sum_{i=1}^{n} \sum_{q' \in R(\xi_i)} f(q')$$

(6)

By redefining the denominator in (6) as $F(q)$:
From formula (7) we can see that the described selection procedure is equivalent to the classical one, provided that the probability of selecting $\psi$ from $A(t)$ be proportional to $P_{\text{sel}}(\psi)$ (The normalization factor, equal for all the $\psi$'s must not be computed).

The advantage of formula (7) is that it reduces the complexity of computing $P_{\text{sel}}(\psi)$ at each selection step to $O(M) + O(NM)$, in contrast to $O(MN^2)$, as in [Deb & Goldberg, 1989].

### 3. Results and Discussion

In this section we present the results obtained by applying the method described in Section 2 to an artificial domain, in which a multimodal concept consisting of four disjuncts has to be learned from a set of positive and negative examples. In order to reduce the run time, with the aim of repeating the experiment many times to collect a statistical evidence, the chosen problem could be described in propositional calculus, with a template of 18 bits to encode inductive hypotheses. According to the a-priori rules, used to assign the class to 500 positive and 500 negative examples, the first disjunct was represented by 312 instances, the second one by 167, the third one by 56 and the fourth by 35. There also exists another definition for the fourth disjunct, strongly overlapped to the second one, which covers 105 examples. Some time REGAL found this solution instead of the intended one. The goal of the experiment was to check the capability of the algorithm of allowing the formation of small disjuncts in presence of much larger ones.

REGAL is a distributed system, consisting of a set of populations (nodes) interconnected by a hypercube network, where the individuals can migrate from a node to another. Moreover, REGAL adopts a long term tunable control strategy, which we called Tories-and-Whigs: this strategy may range between the two extreme cases of learning only one disjunct at a time and all the disjuncts at the same time. More specifically, a whig node is one which changes its goals during the time, i.e., when the best individual $\psi$ in the population remains the same for a given number $g$ of generations, $\psi$ is considered as an achieved partial solution and, hence, the following actions are repeated: (1) the individual $\psi$ is saved in a permanent memory; (2) the instances in the learning set covered by $\psi$ are declared covered; (3) a new population is generated; (4) the genetic algorithm restarts. As the learning instances declared covered do not contribute to the computation of the fitness of the individuals, the genetic algorithm will focus on individuals covering instances not yet covered. On the opposite, tory nodes wait forever for the formation of a stable and good population, possibly covering all the positive instances of the target concept.

By mixing tories and whigs one can expect an increase of the chances of obtaining more general solutions, by easing the problem of letal matings between competing disjuncts, on the one hand, and by giving more chances of crossing genetically different individuals, on the other. If the system reduces to just one whig node, REGAL will learn one disjunct at a time.

First of all, REGAL has been run using populations of different sizes and with different numbers of tory nodes, in order to monitor the capability of the algorithm of allowing the formation of subpopulations. The system was in general able to produce a reasonably approximate solution, but not a perfect one. In particular, the first two disjuncts have always been found, the third one most of the times was found, whereas the smallest one was never found but, in its place, a slightly inconsistent, more general solution has been generated. Using a migration rate of the 5%, the number of nodes and the size of the population did not show any particular influence on the final solution, provided that the global population contained at least 200 individuals and that each node had at least 75 individuals. Using smaller populations, no reasonable solution could be found.

In Table I we compare the solutions obtained for the classical problem of “mushroom classification” using a tory-and-whig strategy with the one, more specific, obtained learning one disjunct at a time.

### 4. Conclusions

From the performed experiments, we can conclude that the task of learning multimodal concepts, or several concepts at the same time, requires more sophisticated approach that those required to learn a unimodal concept. The problem cannot be solved by simply inventing a more complex shared function, but also new evolution strategies are to be designed. The possibility of developing a more theoretical approach, trying to describe an approximate
evolution of the population, is being also investigated, in the hope of obtaining a guidance for the design of more effective population models.

Table I
Rules learned by REGAL for the “mushroom classification” problem. The learning contained 4000 examples. The rules found are complete and consistent also with respect to the whole set of examples in the dataset.

<table>
<thead>
<tr>
<th>with sharing, whigs &amp; tories</th>
<th>Rule cover</th>
<th>Total cover</th>
<th>without sharing, one at a time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule 1</td>
<td>(1859 0)</td>
<td>(1859 0)</td>
<td>Rule 1</td>
</tr>
<tr>
<td>Rule 2</td>
<td>(1727 0)</td>
<td>(1973 0)</td>
<td>Rule 2</td>
</tr>
<tr>
<td>Rule 3</td>
<td>(387 0)</td>
<td>(2000 0)</td>
<td>Rule 3</td>
</tr>
<tr>
<td>Rule 4</td>
<td>(0 1946)</td>
<td>(0 1946)</td>
<td>Rule 4</td>
</tr>
<tr>
<td>Rule 5</td>
<td>(0 187)</td>
<td>(0 1983)</td>
<td>Rule 5</td>
</tr>
<tr>
<td>Rule 6</td>
<td>(0 1139)</td>
<td>(0 2000)</td>
<td>Rule 6</td>
</tr>
</tbody>
</table>

References