RULE LEARNING IN EXPERT SYSTEMS USING GENETIC ALGORITHM: 1, CONCEPTS

Kwong Sak Leung[+], Yee Leung[++], Leo So[+++], and Kin Fai Yam[+]

[+]Department of Computer Science
[++]Department of Geography, and
Center for Environmental Studies
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong
[+++]Computer Science Department
University of Arizona
Tucson, AZ 85721, U.S.A.

Keywords: Genetic algorithm, rule learning, knowledge engineering, token competition, rule migration, expert system.

Abstract

Rules are the essence of a rule-based expert system. However, the acquisition of rules is known to be the bottleneck of the knowledge engineering process. Genetic algorithms (GAs) are investigated as a tool to acquire rules automatically from some training examples. A platform code-named *SCION* is built for the development of GA based applications. Two novel ideas, namely, token competition and rule migration are introduced in this two-part series of papers. Their effects on the efficiency and effectiveness of applying GAs to rule acquisition are explored. From the results obtained so far, GAs seem to be a very promising tool for such an investigation. Basic concepts of *SCION* are introduced in part 1. Case studies and results are presented in part 2.

1. Introduction

The early expert systems (ESs) are mainly rule-based systems. Nevertheless, object oriented ESs are becoming increasingly popular. However, even within an object oriented ES [1]-[2], rules frequently form an essential part of the knowledge base and are one of the most important knowledge representation methods. Unfortunately, acquiring rules from domain experts is not an easy task. On the other hand, it is almost impossible for a knowledge engineer to extract rules from static databases. Rule acquisition thus becomes a bottleneck in the knowledge engineering process.

A genetic algorithm (GA) [3]-[13] is a general search technique that imitates the evolution processes of nature. In order to use a GA to learn rules from examples given as a set of data, an initial population of rules is germinated randomly which are then subjected to *rule evaluation*, a competitive process in which the rules are ranked according to their fitness obtained from comparing with the set of training examples. The weaker rules are eliminated. The remaining elite rules are used by the GA to produce offsprings (new rules) by *crossover* and/or *mutation*. To complete an iteration (an evolution cycle), the new born rules can then join the competition (rule evaluation) after being treated by a *rule tidying* process which prunes redundant components in each new rule. The cycling stops when the population of rules satisfies certain criteria or a preset number of iterations is reached. Of course, a GA can be used for refinement of rule bases.

The crossover operation involves two rules. The randomly selected part from each rule are joined by a random Boolean relation to give new offsprings. For example, (X2 <> X3) AND (NOT (X1 < 4)) can crossover with (X1 + X2 = 9) OR (X4 < 5) to give (NOT (X1 < 4)) OR (X1 + X2 = 9), where OR is randomly generated. The mutation operator randomly selects a candidate from the elite rules and performs a grow (add), slim (cut) or a change operation on one of its randomly selected opcodes or operands.

The next section gives the overall algorithm and architecture of *SCION*. The two novel ideas, *token competition* and *rule migration*, introduced into our rule learning algorithm are detailed in sections 3 and 4 respectively. The paper is concluded with a statement leading to part 2 of this two-part series of paper.

2. Overall Architecture of *SCION*

The overall system flows of *SCION* is depicted in Fig. 1 and is explained in brief in this section.

2.1 Rule Representation

In order to increase efficiency, we employ a new structure, a chain of duples instead of tree representation. A duple is an entity containing two values which are associated with an attribute of a rule. The values are the lower and upper bounds of the attribute. Suppose the rule R1 in class 1 containing three duples has the following form:

$$((4, 7), (3, 9), (11, 20))$$

The equivalent Boolean form is:

$$\text{IF } (4 \le X_1 \le 7) \text{ AND } (3 \le X_2 \le 9) \text{ AND } (11 \le X_3 \le 20) \text{ THEN CLASS} = 1.$$

The advantage of using duples is that a greater resemblance between the rule structure and its biological counterpart can be established. The chain of duples is an analogy of a chromosome and a duple is an analogy of a gene. Also, with duple representation, genetic operators like crossover and mutation can be made faster and simpler because the simple array data structure storing the duples resembles the gene-chromosome relationship. A crossover operator involves only the splitting of two parents and then the recombination of their two separated sections. The mutation operator involves only the random changing of an arbitrary duple. Thus the learning process can be sped up.

The simple representation only allows AND and ≤ relationships in a rule, but almost all relationships can be simulated by a set of simple rules. For example, an OR relationship can be modeled by 2 separate rules in the inference process.

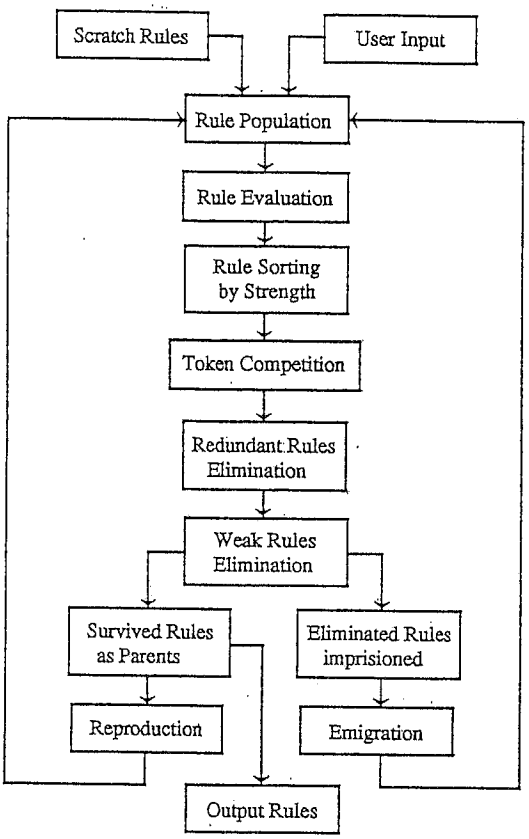

Fig. 1 The overall system flows of *SCION*

2.2 Rule Evaluation and Sorting

This module is used to calculate the strength of all rules by keeping track of their hit scores and false alarms as follows:

```
for each class i
   read sample data until end of file
      for each rule in class i
         if it can classify the data
            increase its hit score by 1
      for each rule in other class
         if it can classify the data
            increase its false alarm by 1
```

All rules within one class are sorted in a decending order by their strengths.

A class can consist of two data zones with contrasting sizes. There is a risk that the smaller zone will be ignored. It is because any rule landing on the larger zone will enjoy a good score (HIT-FALSE) even though they may have false alarms. Those rules landing on the smaller zone will suffer from limited credit even though they have no false alarm at all. Then these unfortunate rules will face the risk of being expelled since competition for survival is keen. As a result, this small zone will be left out. Consequently, perfect performance can never be reached.

However, under the HIT/FALSE instead of the conventional HIT-FALSE scoring strategy, rules in small clusters still have the chance to get high credit if they have proportionally less false alarms than those rules in bigger clusters. Therefore, the small clusters are still accounted for. To avoid dividing by zero error when FALSE = 0, a strength adjustment is added to the denominator, making the formula to be HIT/(FALSE + ADJUST).

2.3 Token Competition

This module determines the number of tokens each rule can get. The rules are assumed sorted already. Token allocation is determined as follows:

```
for each class i
  read sample data until end of file
     give token to the first rule in class i classifying it
```

(see section 3 for details)

2.4 Redundant and Weak Rule Eliminations

After token competition, each rule will get its own tokens. Those cannot get any token are classified as redundant rules which are to be eliminated and imprisoned. If the size of the rule set still exceeds the allowed quota of parent rules after redundant rule elimination, the excess rules (the weaker ones) will be eliminated.

2.5 Rule Migration

After rule elimination, the weaker rules eliminated from the population are imprisoned. According to their hits and false alarms to other classes, their scores toward other foreign classes are computed. If any bad rule of a particular class has a score better than the average score of any other class, it will be copied to that class as its immigrant.

The status of the new immigrant is just like an offspring from reproduction, i.e. they are both used to fill up the difference between the total population and the survived parents. Since rule migration happens earlier than rule reproduction, the more immigrant rules migrate to a class, the less offsprings can the original rules in that class reproduce. So an immigration quota is required to set the maximum number of immigrants to a class. (See section 4 for details).

2.6 Reproduction

After two severe screening procedures, all the rules survived will become potential parents. Crossover operation is performed first. Two parents are involved in contributing their "genes" to form the children rules. The selection of parents is a process by which a rule with greater strength has a greater chance of being selected. The quota of reproducing children rules by the crossover operation is determined by the cross ratio supplied by users. The rest of the space belongs to the mutation operation.

2.6.1 Crossover

The selected parent rules will duplicate themselves first. Then each copy will be splitted into two sections with the cut-off point randomly selected. The section before the cut-off point is detached from the rest of the body. Then these sections will be exchanged and recombination takes place, giving two new children rules. Suppose the cut-off point is at the 4th position for the following two copies of parent rules:

```
((1, 10), (4, 11), (20, 30))
((58, 90), (7, 40), (1, 5))
```

Then, two children rules are obtained as follows:

```
((1, 10), (4, 40), (1, 5))
((58, 90), (7, 11), (20, 30))
```

Each recombination must be checked to prevent inappropriate matching between a large lower bound and a small upper bound. Also the children rule produced must not resemble any existing rules. If any duplication is found, the crossover operator is

then reapplied again until a successful mating is achieved or until too much failure is encountered. The latter case may suggest that the combinations of genes of the parents are exhausted. Then the excess quota will be granted to the mutation operation.

2.6.2 Mutation

The mutation operation will just select randomly a value to be changed. The partner of the value in the duple will be referenced to guarantee that no lower bound is greater than the upper bound. Suppose mutation takes place at position 2 of the following rule:

$$((4, 23), (17, 34), (1, 9))$$

The child rule is then obtained as:

$$((4, 23), (17, 91), (1, 9))$$

3. Elaboration on Token Competition

The inbreed mating poses a great threat to the robustness of the GA. The problem can be illustrated by the following simple example.

Suppose the problem space is described by two attributes: X_1, X_2. It can be portrayed as a two dimensional diagram in Fig. 2.

The shaded area is the location of sample data of a particular class. Take it as class 1. Also suppose there are two rules which can correctly classify this zone of data.

Although both rules overwhelm the sample zone a little bit, the overspill does not cause any false alarm because no other class of the sample data is covered.

In Fig. 2, the rules expressed in Boolean logic form are: R1: ((b, a), (s, t)), and R2: ((d, c), (u, v)). Once they happen to mate each other, their offspring may take the form of: C1: (b,c) (u,v)), which also embraces the same data zone.

As this phenomenon propagates, the whole population will gradually degenerate into a set of homogeneous rules. Then the evolution will fall into a vicious circle as the homogeneous parent rules reproduce homogeneous children rules.

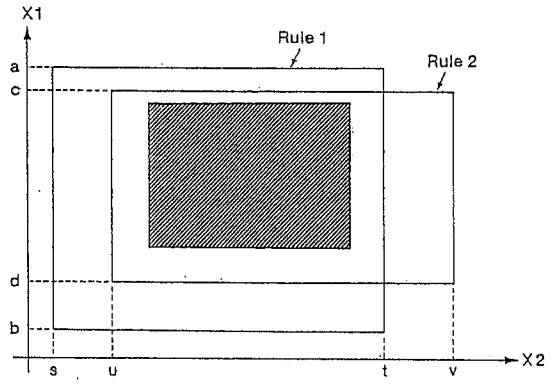

Fig. 2 GA with two redundant rules

Token Competition is designed to remove this hindrance. In this mechanism, each sample datum is regarded as a token. Once a rule can correctly classify this datum, it will seize this token so that rules to follow cannnot get it. After the whole sample database is parsed, those rules with no token at hand will be killed. The priority of receiving tokens is determined by the strength of the rules. Strong rules are encouraged to seize as many tokens as they can, leaving those weak rules starving. As a result, the whole sample data set can be represented by only a small set of strong rules, making the final production rules more concise. Besides, more room can be saved for reproducing more children rules by cutting those redundant parent rules, without affecting the performance. The robustness of the GA is enhanced as more search points are explored by this larger set of children rules.

Fig. 3 is an example of token competition. Since R1 embraces the whole shadow, it is the strongest. It will just sweep the token pools and cause the weaker rules R2 and R3 starving to death. The beauty of token competition stems from its simplicity.

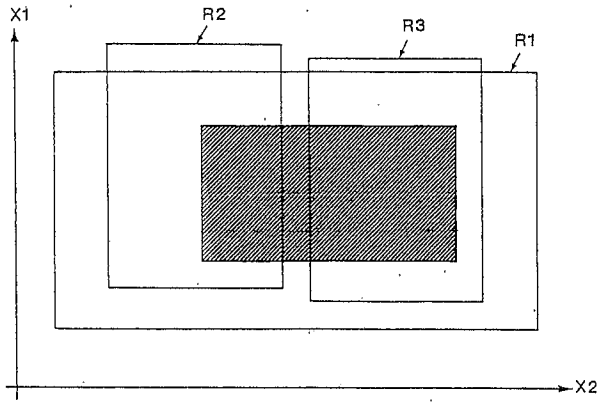

Fig. 3 GA with token competition among three rules

4. Elaboration on Rule Migration

Rule Migration is a novel idea in GAs. The concept behind this idea is simple. During the evolution process, a weak species in one class may be a strong species in another class and can be preserved rather than simply discarded.

This idea comes from the observation that the classes themselves are disjoint. Therefore a rule can only score high in its should-be class. Thus the good rule in a certain class need not migrate to other classes, for it must have low scores in other classes.

However, a poor rule in one class may score well in other classes. Without migration, this valuable rule for other classes may be discarded and wasted.

For example in Fig. 4, assume that the two-dimensional problem space has two attributes X_1, X_2. Four areas I, II, III and IV on this space correspond to classes I, II, III and IV respectively. They are disjoint.

Assume at a certain time in the rule population of class I, a rule (R1) is somehow reproduced as an offspring. Clearly, R1, with so little hit scores and so many false alarms, cannot survive in class I. However, it *may* be valuable to classes IV and II. So it *can* migrate to both IV and II. (Actually, if the original rules in these two classes are very strong, or if the quotas for the destination classes are already filled by some other eligible alien rules, then R1 cannot migrate to them.)

Another rule (R2) is an offspring reproduced from class II. If there are already many strong rules in class II, this new rule may be useless to it. However, it is valuable to class III and can migrate to class III. It cannot migrate to classes I and IV as R2 hits nothing in these two classes.

The immigrant rules can be treated as offsprings in the reproduction process. If after migration, there is still room for crossover and mutation, original rules in that class will reproduce to fill

up the population size.

If an immigrant rule behaves better than the newly produced offsprings, then its last immigration is said to be successful. Otherwise, it has less contribution to this class than the offsprings from the original population. So after one generation, it will be discarded.

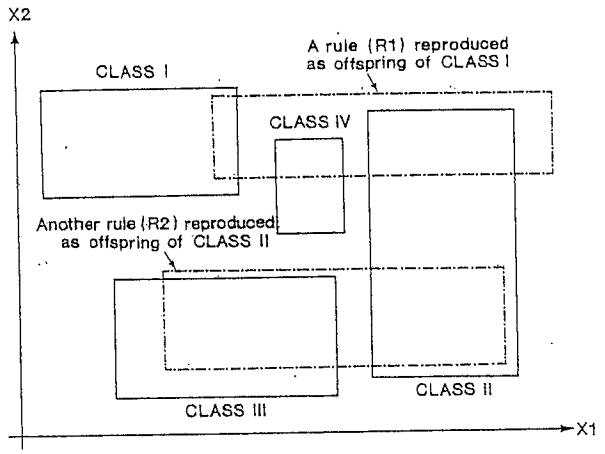

Fig. 4 GA with rule reproduction and migration

5. Conclusion

We have outlined in brief the basic concepts in *SCION*, a platform for developing GA based applications. In addition to basic GA features, two novel ideas, *token competition* and *rule migration*, have been proposed and incorporated into *SCION*. Their plausible contributions to the improvement of GAs have been discussed. Some empirical findings are presented in part 2 of this two-part series of papers to substantiate the theoretical arguments.

References

[1] K.S. Leung, Y. Leung, and M.H. Wong, "The Integration of Rule-based and Procedural Methods to Solve Optimization Problems through Expert-system Technology," in J-L Verdegay and M. Delgad (eds.), "The Interface between Artificial Intelligence and Operations Research in Fuzzy Environment;" Verlag TÜV Rheinland, 1989.

[2] K.S. Leung and M.H. Wong, "An Expert System Shell using Structured Knowledge — an Object-oriented Approach," IEEE Computer, Vol. 23, pp. 38-47, 1990.

[3] L.B. Booker, D.E. Goldberg and J.H. Holland, "Classifier Systems and Genetic Algorithms," Artificial Intelligence, Vol. 40, pp. 235-282, 1989.

[4] Lawrence Davis, "Genetic Algorithms and Simulated Annealing," BBN Laboratories, Cambridge, Massachusetts, 1987.

[5] Lawrence Davis, "Adapting Operator Probabilities in Genetic Algorithms," BBN Laboratories, Cambridge, Massachusetts, 1989.

[6] K.A. De Jong, "Analysis of the Behaviour of a class of Genetic Adaptive Systems," Ph.D. thesis, University of Michigan, Ann Arbor, 1975.

[7] Richard Forsyth, "Beagle — a Darwinian approach to pattern recognition," Kybernetes, Vol. 10, pp. 159-166, 1981.

[8] R.B. Hollstien, "Artificial Genetic Adaptation in Computer Control Systems," Ph.D. thesis, University of Michigan, Ann Arbor, 1971.

[9] Chuck Karr, "Genetic Algorithms for Fuzzy Controllers," AI EXPERT, pp. 26-33, February 1991.

[10] David J. Montana, "Empirical Learning Using Rule Threshold Optimization for Detection of Events in Synthetic Images," Machine Learning, Vol. 5, pp. 427-450, 1990.

[11] David Shaffer and Amy Morishima, "An Adaptive Crossover Distribution Mechanism for Genetic Algorithms," in John J. Grefenstette (ed.), "Genetic Algorithms and their Applications," Lawrence Erlbaum Associates, 1987.

[12] Gilbert Syswerda, "Uniform Crossover in Genetic Algorithms," BBN Laboratories, Cambridge, Massachusetts, 1989.

[13] Darrell Whitley, "Using Reproductive Evaluation to Improve Genetic Search and Heuristic Discovery," in John J. Grefenstette (ed.), "Genetic Algorithms and their Applications," Lawrence Erlbaum Associates, 1987.