SUPPLEMENT TO KYBERNETIKA VOLUME 28 (1992), PAGES 33–36

ON LEARNING IN A FUZZY RULE-BASED EXPERT SYSTEM

ANDREAS GEYER–SCHULZ

The main motivation fo adding learning capabilities to fuzzy rule-based expert systems is the desire to reduce the cost and time of knowledge acquisition. Particularly attractive are in this context learning algorithms with the ability to synthesize rules from past cases already available in the data-base. With the aim of automating knowledge acquisition we present in this article fuzzy classifier systems which integrate a fuzzy rule base, a genetic algorithm and an apportionment of credit function. As a first result we give a variant of the mutation operator which allows us to derive a.global convergence result for the genetic algorithm under rather weak assumptions. With this approach a combination of the advantages of genetic algorithms and simulated annealing algorithms is achieved.

1. A FUZZY CLASSIFIER SYSTEM

A classifier system is a machine learning system which learns rules to guide its performance in an arbitrary environment [4]. Its main components are a production system, an apportionment of credit function and a genetic algorithm.

A *fuzzy* classifier system learns rules written in a *fuzzy* rule language and uses a *fuzzy* production system. Although there is a wide range of fuzzy production systems differing in their syntax and semantics [3], we concentrate in the following on fuzzy production systems with a very simple syntactic structure. The reasons for this are twofold. First, these production systems have been successfully applied to various control problems in robotics and industrial automation (e.g. [1], [7], [8]). Secondly, the simple syntactic structure of the rule language and the small number of rules for a rule-base are characteristics common to fuzzy control applications which make these problems amenable to genetic machine learning. Consider for example the fuzzy rule language $FL = \{V_T, V_N, P, S\}$ used in [8] with $V_T =$ { "IF", "IS", "AND", "THEN", "SL", "ΔM", "ΔU", "POSITIVE BIG", "POSITIVE SMALL", "ZERO", "NEGATIVE SMALL", "NEGATIVE BIG" } and $V_N =$ { ⟨rule base⟩, ⟨rule⟩, ⟨variable⟩, ⟨label⟩} and $S =$ ⟨rule base⟩ and the following productions P:

⟨rule base⟩ := ⟨rule⟩ | ⟨rule⟩ ⟨rule base⟩

⟨rule⟩ := "IF" ⟨variable⟩ "IS" ⟨label⟩ "AND" ⟨variable⟩ "IS" ⟨label⟩ "THEN" ⟨variable⟩ "IS" ⟨label⟩

⟨variable⟩ := "SL" | "Δ M" | "Δ U"

⟨label⟩ := "POSITIVE BIG" | "POSITIVE SMALL" | "ZERO" | "NEGATIVE SMALL" | "NEGATIVE BIG"

The genetic algorithm mimicks natural evolution in order to learn a rule-base which performs well with regard to the objective function of the system. Instead of a single current rule-base the genetic algorithm maintains a population of rule-bases as current solution. With a selection rule the genetic algorithm generates the next generation from the current population. A simple selection rule of a genetic algorithm is composed of the "genetic" operators reproduction, crossover and mutation, but several advanced operators like for example dominance with diploid or polyploid chromosoms, inversion and reordering, duplication and deletion, sexual determination and differentiation have been studied [4]. In Section 2 we introduce a selection rule based on a special mutation operator with the aim of ensuring global convergence and in Section 3 we prove the global convergence for (fuzzy) classifier systems with the selection rule given in Section 2.

2. A "MONTE CARLO" SELECTION RULE

Definition 1. A "Monte Carlo" selection rule for (fuzzy) classifier systems generates the rule-bases which form the population of the next generation in the following way:

1. Generate c children by applying a suitable mix of genetic operators to the p parent individuals in the current population.
2. Select the best of all the $p + c$ individual rule-bases as the first rule-base in the next population.
3. Generate the second rule-base by the "Monte Carlo" mutation operator given below.
4. Select $p - 2$ rule-bases from all the $p + c$ rule-bases by any rule you like.

Definition 2. A "Monte Carlo" mutation operator selects a rule-base from the set R of all rule-bases which can be generated in a language L with at most d derivations with probability $P = \frac{1}{card(R)}$.

Rule 2 of Definition 1 is from R. F. Hartl [6]. Several variants of rule 4 Definition 1 can be found in [6].

The "Monte Carlo" mutation operator essentially generates a large neighbourhood for each rule-base. For each rule-base r of R it generates a neighbourhood $N(r)$ for which $N(R) = R$ holds. From a rule-base r of R any other rule-base x of R is reachable with equal probability.

Although we do not know an efficient algorithm for implementing a "Monte Carlo" mutation operator for arbitrary languages, for the fuzzy rule-languages used in fuzzy-control like FL, however, we are able to give efficient implementations, since the number of different rules is still small (e. g. 3375 different rules in FL).

3. GLOBAL CONVERGENCE

We say "the optimal rule-base has been learned by a genetic algorithm" if at least one global optimum is in the current population.

Theorem 1. A genetic algorithm with "Monte Carlo" selection rule learns the optimal rule-base in language L with respect to the apportionment of credit function f in n generations with the probability of failure tending to zero as n tends to ∞. (*Global Convergence*)

We establish the proof of Theorem 1 with the help of Assumption 1 and Lemma 1.

Assumption 1. The set R of possible rule-bases in the rule-language L is assumed to be of finite cardinality. (*Finite Cardinality*)

We can easily enforce Assumption 1 for every formal language L by restricting the number of possible derivations to d. The set R of all language expressions which can be generated from the start symbol S and which contain only terminal symbols by using less or equal to d derivations is then obviously of finite cardinality. For practical applications it suffices to fix d. In fact, for any computer implementation Assumption 1 holds, since computers have only limited memory.

Lemma 1. The probability of reaching an optimal element in a set R with finite cardinality in k generations is larger than $P \geq (\frac{1}{card(R)})^k$. (See [2].)

To prove Lemma 1 we slightly rephrase Dantzig [2].

Proof. The probability of making a selection which leads to an optimal element is at least $\frac{1}{card(R)}$, since we choose with equal probability. Hence for n generations Lemma 1 holds. Moreover, the probability to reach an optimum before k generations is less than $1 - (\frac{1}{card(R)})^k$. It follows that the probability of failing to reach an optimum by $2k$ generations is less than $(1 - (\frac{1}{card(R)})^k)^2$ and the probability of failing to reach an optimum by $n = t.k$ generations is less than $(1 - (\frac{1}{card(R)})^k)^t$. This expression, however, tends to zero as $t \to \infty$. □

Note, that because of rule 2 of Definition 1 a global optimum will always remain in the population, as soon as it is found.

Until very recently no global convergence proof for genetic algorithms like the result of Hajek for simulated annealing [5] seemed to be available. In [6] R. F. Hartl proves the global convergence of genetic algorithms with a selection rule which uses a standard mutation operator instead of a "Monte Carlo" mutation operator. The proof is essentially based on the assumption that the Markovian chain described by the selection rule is irreducible. In our variant of the global convergence proof Hartl's assumption always holds because of the very construction of the "Monte Carlo" mutation operator. Since we are mainly interested in fuzzy classifier systems, the "Monte Carlo" mutation operator has been tailored to feasible solution sets which are derived from the generative grammar of a fuzzy rule language.

4. SUMMARY

In this paper we have presented a fuzzy classifier system based on a Monte Carlo selection rule. By making only the weak assumption of finite cardinality of the number of feasible rule-bases we have been able to derive a global convergence result for classifier systems. Furthermore, we have shown that fuzzy rule-bases as for instance those used in fuzzy control are suited for genetic machine learning systems like fuzzy classifier systems because of the restricted syntax of the fuzzy rule languages used and because of the small number of rules used for one application. For practitioners the main advantage of the algorithm is that no expertise is required to apply it.

REFERENCES

[1] T. Akiyama and T. Sasaki: Extended fuzzy traffic control model on urban expressway. Preprints of Second IFSA Congress *1* (1987), 1, 278 – 279.

[2] G. B. Dantzig: Linear Programming and Extensions. 6th Edition. Princeton University Press, Princeton, NJ 1963.

[3] A. Geyer-Schulz: Fuzzy rule-based expert systems. In: APL Techniques in Expert Systems (J. R. Kraemer and P. C. Berry, eds.), ACM SIGAPL, Syracuse, NY 1988.

[4] D. E. Goldberg: Genetic Algorithms in Search, Optimization & Machine Learning. Addison Wesley, Reading, MA 1989.

[5] B. Hajek: Cooling schedules for optimal annealing. Math. Oper. Res. *13* (1988), 13, 311 – 329.

[6] R. F. Hartl: Global Convergence Proof for a Class of Genetic Algorithms. Unpublished Manuscript.

[7] K. Hirota, Y. Arai and S. Hachisu: Real time pattern recognition and fuzzy controlled robot-arm. Preprints of Second IFSA Congress *1* (1987), 1, 274 – 277.

[8] D. Yasuhiko and M. Honma: Improved fuzzy-set temperature distribution control for electric furnace. Preprints of Second IFSA Congress *1* (1987), 1, 270 – 273.

Dr. Andreas Geyer-Schulz, Department of Applied Computer Science, Institute of Information Processing and Information Economics, Vienna University of Economics and Business Administration, Augasse 2-6, A-1090 Vienna. Austria.