GENERATING FUZZY RULES FROM EXAMPLES USING GENETIC ALGORITHMS*

Francisco HERRERA, Manuel LOZANO, Jose Luis VERDEGAY

Dept. of Computer Science and Artificial Intelligence
University of Granada,
18071 - Granada, Spain
e-mail: fherrera@ugr.es, lozano@robinson.ugr.es, jverdegay@ugr.es

ABSTRACT

The problem of generation desirable fuzzy rules is very important in the development of fuzzy systems. The purpose of this paper is to present a generation method of fuzzy control rules by learning from examples using genetic algorithms. We propose a real coded genetic algorithm for learning fuzzy rules, and an iterative process for obtaining a set of rules which covers the examples set with a covering value previously defined.

Keywords: Fuzzy rules, learning, genetic algorithms.

1. Introduction

Fuzzy rules based systems have been shown to be an important tool for modeling complex systems, where due to the complexity or the imprecision, classical tools are unsuccessful.

In [19, 5] it was proved that fuzzy systems are universal approximators in the sense that for any continuous systems is possible to find a set of fuzzy rules able of approximating it with arbitrary accuracy. The problem is how to find the rules.

There are different modes to derive them:

- Based on Expert Experience and Control Engineering Knowledge.
- Based on Operator's Control Actions.
- Based on the Fuzzy Model of a Process.
- Based on Learning or Self organizing Control.

The construction of fuzzy rules has been mainly based on the operator's control experience or actions. Unfortunately, acquiring rules from experts is not an easy task, and on the other hand, it is very difficult for a knowledge engineer to extract rules from static databases. Thus, rule acquisition becomes a bottleneck in the knowledge engineering process.

Recently there is an extensive literature of learning from examples in a fuzzy environment and various methods have been proposed. These methods include the learning methods employing Descent Method, [1], or Neural Network, [21, 11], or fuzzy discretization and clustering techniques [14, 22], or mountain clustering method, [20], or frequencies in the identification of a system, [6], or Belief or Uncertainty measures, [3, 15, 7], as well as tools to identify the structure of a fuzzy model [18, 16, 17].

In a learning process we can distinguish different components:

1. A generation method of desirable fuzzy rules able to include the complete knowledge of the set of examples.
2. A simplify method:
   2.1. Resolving conflicts among the generated rules,
   2.2. Finding the final set of fuzzy rules able to approximate the input-output behavior of a real system.
3. A tuning method of the set of rules obtained in the step 2.

* Proceeding Fifth International Conference of Information Processing and Management of Uncertainty in Knowledge-Based Systems, Paris July 4-8, 1994, 675-680.This research has been supported under proyect PB92-0933.

We will focus this paper on the description of a method for generating desirable fuzzy rules from examples using genetic algorithms. We propose a generation method of fuzzy rules by means of a special real coded genetic algorithm (RCGA), where a chromosome represents a fuzzy rule, and is evaluated by means of a frequency method ([6, 7]). The RCGA will find good rules, and with an iterative process we obtain a set of rules covering the set of examples.

In order to that, we set out the paper as follows. Next section describes the fuzzy rule structure that we use. Section 3 introduces the genetic algorithms, and section 4 presents the generation method of fuzzy rules.

2. Fuzzy Rule Structure

We consider rules with a free structure, without an initial referential fuzzy sets.

The rules have the form

R_i : IF x_1 is A_{i1} and ... and x_n is A_{in}
THEN y_1 is B_{i1} and ... and y_m is B_{im}

where $x_1, ..., x_n$, and $y_1, ..., y_m$ are variables representing the process state variables and the control variables respectively; and $A_{i1}, ..., A_{in}$, $B_{i1}, ..., B_{im}$ are fuzzy sets in the universes of discourse $U_1, ..., U_n$, $V_1, ..., V_m$.

These fuzzy sets are characterized by their membership functions

$$A_{ij}(B_{ih}) : U_j(V_h) \rightarrow [0, 1],$$
$$j = 1, ..., n, \; h = 1, .., m$$

We consider trapezoidal membership functions with parametric representation achieved by means of the 4-tuple $(a^1_{ij}, a^2_{ij}, a^3_{ij}, a^4_{ij})$, $(b^1_{ih}, b^2_{ih}, b^3_{ih}, b^4_{ih})$.

3. Genetic Algorithms

Genetic Algorithms (GA) are search algorithms that use operations found in natural genetics to guide the trek through a search space. GA are theoretically and empirically proven to provide robust search in complex spaces, giving a valid approach to problems requiring efficient and effective search [8].

Although there are many possible variants of the basic GA, the fundamental underlying mechanism operates on a population of chromosomes or individuals (representing possible solutions to the problem) and consists of three operations:

(1) evaluation of individual fitness,

(2) formation of a gene pool, and

(3) recombination and mutation.

The following figure shows the structure of a simple GA,

Procedure genetic algorithm
begin (1)
  $t = 0$;
  initialize $P(t)$;
  evaluate $P(t)$;
  While (**Not** *termination-condition*) **do**
  begin (2)
    $t = t + 1$;
    select $P(t)$ *from* $P(t-1)$;
    recombine $P(t)$;
    evaluate $P(t)$;
  end (2)
end (1)

Figure 1: Structure of a GA

Thus, it is generally accepted that any GA to solve a problem must take account the five following components:

1. *A genetic representation of solutions to the problem,*
2. *a way to create an initial population of solutions,*
3. *an evaluation function which gives the fitness of each individual,*
4. *genetic operators that alter the genetic composition of children during reproduction, and*
5. *values for the parameters that the GA uses (population size, probabilities of applying genetic operators, etc.).*

4. Generating Fuzzy Rules

Let us suppose we know a set of p training examples $E_p = \{e_1, ..., e_p\}$ of the system consisting of the values that the variables take during an experiment in which the system is controlled by an expert:

"in the time t = k, the value of the variable vectors X and Y are ex^k and ey^k respectively"

If we wish to generate a set of rules describing the behavior of a system, then a condition over the set of generated rules, R, is necessary to establish. This is the requirement

of covering all possible situation-action pairs, $e_k \in E_p$, the completeness property. This may be formalized for some constant $\tau \in [0,1]$, and requires the non-zero union of fuzzy sets $A(\cdot)$, $B(\cdot)$, $i = 1, ..., T$, $|R| = T$,

$$C_R(e_k) = \bigcup_{i=1..T} R_i(e_k) > \tau \quad k = 1, ..., p$$

$$R_i(e_k) = *(A_i(ex^k), B_i(ey^k))$$

$$A_i(ex^k) = *(A_{i1}(ex_1^k), ..., A_{in}(ex_n^k))$$

$$B_i(ey^k) = *(B_{i1}(ey_1^k), ..., B_{im}(ey_n^k))$$

where $*$ is a t-norm, and $R_i(e_k)$ is a compatibility degree between the rule R_i and the example e_k .

We propose to depth in this requirement: Given a set of rules R, we define the *covering value* of a example e_k with a base of rules R as

$$CV_R(e_k) = \sum_{i=1}^{T} R_i(e_k)$$

and we will require in our method the following condition

$$CV_R(e_k) \geq \epsilon \quad k = 1, ..., p$$

with $\epsilon = 1$ as an initial proposal.

Next, we propose a generation method of rules by means of a RCGA and a covering method of the set of examples.

Every chromosome of the RCGA represents a fuzzy rule, and this is evaluated by means of the concepts of frecuency, possitive and negative examples. Then, the RCGA will obtain a good rule with a) a great frecuency value, b) a big set of possitive examples with compatibility degree greatter than τ, and c) without negative examples.

The covering method will be an iterative process which runs the RCGA over the set of examples, choices the best chromosome (rule), asigns to every example the relative covering value, and removes the examples with a covering value greater than ϵ from the set of examples.

First we describe the components of the RCGA, and fatherly we present the covering method.

4.1. Real Coded Genetic Algorithms

In order to describe the RCGA we present the five components indicated in section 3.

1. *Representation*

   In the population of the RCGA a candidate solution C_r, $r = 1, ..., M$, represents a fuzzy rule

   IF x_1 is A_{r1} ... and x_n is A_{rn}
   THEN y_1 is B_{r1} and ... and y_m is B_{rm}

   where the real values a_{rj}^1, a_{rj}^2, a_{rj}^3, a_{rj}^4, b_{rh}^1, b_{rh}^2, b_{rh}^3, b_{rh}^4 characterize the membership functions of A_{rj} and B_{rh} $j = 1, ..., n$, $h = 1, ..., m$, respectively.

   Thus, C_r codes the vector values:

   $$(a_{r1}^1, a_{r1}^2, a_{r1}^3, a_{r1}^4, ..., a_{rn}^1, a_{rn}^2, a_{rn}^3, a_{rn}^4,$$

   $$b_{r1}^1, b_{r1}^2, b_{r1}^3, b_{r1}^4, ..., b_{rm}^1, b_{rm}^2, b_{rm}^3, b_{rm}^4)$$

   As was justified in [9,10], we propose to approach this problem with real coded genes together special genetic operators developed for them. Then a rule would be a chromosome vector coded as a vector of floating point numbers. The precision of such approach depends on the underlying machine, but is generally much better than the binary representation.

   Finally, we represent a population of M chromosomes (rules) by C, and it is set up as follows:

   $$C = (C_1, ..., C_M).$$

   Now, the fundamental underlying mechanisms are developed.

2. *Formation of an initial population or gene pool*

   The domain of every input variable X_j is a close real interval $U_j = [a_j, b_j]$. Similarly, the domain of the output variable Y_h is the real interval $V_h = [c_h, d_h]$.

   The initial gene pool is created partially from $E_t \subseteq E_p$ (t chromosomes) and the rest randomly ($M - t$ chromosomes):

   Let $t = \min\{|E_p|, M/2\}$, then we choice randomly t examples from E_p , and for every one we determine the chromosome (rule) belonging to the initial gene pool as follows:

   Suppose the example $e_k \in E_t$ and the component $ex_j^k \in [a_j, b_j]$, $\triangle ex_j^k = \min\{ex_j^k -$

$a_j, b_j - ex_j^k\}$, let $\delta(ex_j^k)$ be a random value in the range $[0, \triangle ex_j^k]$, then we form the membership function by the 4-tuple

$$(ex_j^k - \delta(ex_j^k), ex_j^k, ex_j^k, ex_j^k + \delta(ex_j^k)).$$

The procedure is the same for the rest components of e_k.

The rest $M - t$ chromosomes of the initial population are chosen randomly, every gene in its respective interval,

$$C_r = (c_{r1}, ..., c_{rl})$$

$l = 4(n + m)$, with requirements

$$c_{4s+1} \leq c_{4s+2} \leq c_{4s+3} \leq c_{4s+4},$$

$$s = 0, ..., n + m - 1.$$

3. *Evaluation of individual fitness*

The frequency of any fuzzy rule R_i through the set of examples E_p is defined as ([6, 7])

$$\Psi_{E_p}(R_i) = \frac{\sum_{k=1}^{p} R_i(e_k)}{p}$$

with $R_i(e_k)$ the compatibility degree.

For a fuzzy rule R_i we define:

a) The set of the positive examples to R_i with compatibility degree greater than τ as:

$$E^+(R_i) = \{e_k \in E_p / R_i(e_k) > \tau\}$$

with

$$n_{R_i}^+ = |E^+(R_i)|,$$

$$G_{R_i} = \sum_{e_k \in E^+(R_i)} R_i(e_k)/n_{R_i}^+.$$

b) The set of the negative example to R_i as:

$$E^-(R_i) = \{e_k \in E_p / R_i(e_k) = 0 \text{ and } A_i(ex_k) > 0\}$$

with

$$n_{R_i}^- = |E^-(R_i)|.$$

An example is considered negative for a rule when it matches better with some other rule having the same antecedent but different consequent.

An evaluation function to the rule R_i and therefore a fitness function to the associated chromosome C_i can be defined as:

$$F(C_i) = \begin{cases} 0 & \text{if } n_{R_i}^- > 0 \\ g(n_{R_i}^+, \Psi_{E_p}(R_i), G_{R_i}) & \text{otherwise} \end{cases}$$

with g an increasing function in its variables, with the objective of making maximum the fitness function.

4. *Genetic Operators*

During the reproduction phase of the GA we use two classical genetic operators, mutation and crossover. We propose to use the non-uniform mutation proposed by Z. Michalewicz, [12]. If $C_v^t = (c_1, ..., c_k, ..., c_H)$ is a chromosome and the element c_k was selected for this mutation (the domain of c_k is $[c_{kl}, c_{kr}]$), the result is a vector $C_v^{t+1} = (c_1, ..., c_k', ..., c_H)$, with $k \in 1, ..., H$, and

$$c_k' = \begin{cases} c_k + \triangle(t, c_{kr} - c_k) & \text{if } a = 0, \\ c_k - \triangle(t, c_k - c_{kl}) & \text{if } a = 1, \end{cases}$$

where a is a random number which can have a value of zero or one, and, the function $\triangle(t, y)$ returns a value in the range $[0, y]$ such that the probability of $\triangle(t, y)$ being close to 0 increases as t increases:

$$\triangle(t, y) = y(1 - r^{(1 - \frac{t}{T})^b})$$

where r is a random number of the interval $[0, 1]$, T is the maximum number of generations and b is a parameter chosen by the user, which determines the degree of dependency with the number of iterations.

This property causes this operator makes a uniform search into the initial space when t is small, and very locally at later stages.

In relation with the crossover operator, we use the max-min-arithmetical crossover proposed by Herrera et al. [9]. If $C_v^t = (c_1, ..., c_k, ..., c_H)$ and $C_w^t = (c_1', ..., c_k', ..., c_H')$ are to be crossed, we generate:

$$C_1^{t+1} = aC_w^t + (1 - a)C_v^t$$

$$C_2^{t+1} = aC_v^t + (1 - a)C_w^t$$

$$C_3^{t+1} \text{ with } c_{3k}^{t+1} = \min\{c_k, c_k'\}$$

$$C_4^{t+1} \text{ with } c_{4k}^{t+1} = \max\{c_k, c_k'\}$$

5. *Parameters*

   We propose to carry out the experiments with the following parameters:

   - Population size: 60
   - Elitist selection and linear ranking [2]
   - Probability of crossover: P = 0.6
   - Max-Min-Arithmetical crossover, a = 0.35
   - Probability of mutation: Probability of chromosome update: P = 0.6
   - Non-uniform mutation, $b = 5$
   - Stop conditions: N iterations.

4.2. Covering method

Using the above RCGA, the process would be as follows:

1. Initialization:
   - To introduce τ and ϵ.
   - To assign $CV[k] \leftarrow 0$, $k = 1, ..., p$.
2. Over the set of examples E_p we apply the specific RCGA.
3. We select the best chromosome C_r with R_r the associated fuzzy rule.
4. We introduce R_r in the set of rules R, which initially is empty.
5. We obtain $E^+(R_r)$ from E_p .

   For every $e_k \in R_r$ do

   $$CV[k] \leftarrow CV[k] + R_r(e_k),$$

   If $CV[k] \geq \epsilon$ then remove it from E_p .
6. If $E_p = \emptyset$ then Stop else return to Step 2.

5. Conclusions

We have focused this paper on the description of a method for generating desirable fuzzy rules from examples using a RCGA and a covering process of a set of examples. This is the first component of the learning process described in section 1.

About the other components, a first proposal of a simplify method to find the final set of fuzzy rules able to approximate the input-output behavior of a real system is described in [4] (section III.3), and a tuning method of the set of rules is described in [9], both using GA.

REFERENCES

[1] Araki, S., Nomura H., Hayashi, I. and Wakami, N. (1991) A self-generating Method of Fuzzy Inference Rules, Proceedings of IFES'91, 1047-1058.

[2] Back, T., and Hoffmeister, F. (1991) Extended Selection Mechanisms in Genetic Algorithms. Proc. of the Fourth Int. Conf. on Genetic Algorithms, Morgan Kaufmmann, San Mateo, 92-99.

[3] Binaghi, E. (1991) Learning of uncertainty classification rules in medical diagnosis in Symbolic and Quantitative Approaches to Uncertainty, Lectures Notes in Computer Science 548 (R. Kruse, P. Siegel, Eds.), 115-119.

[4] Castro, J.L., Delgado, M., and Herrera, F. (1993) A learning method of fuzzy reasoning by genetic algorithms. Proc. First European Congress on Fuzzy and Intelligent Technologies, Aachen, 804-809.

[5] Castro, J.L. (1993) Fuzzy Logic Controllers are Universal Approximators. To appear in IEEE Transactions on Systems, Man, and Cybernetics.

[6] Delgado, M. and González, A. (1993) An Inductive Learning Procedure to Identify Fuzzy Systems. Fuzzy Sets and Systems 55, 121-132.

[7] Delgado, M. and González, A. (1993) A frequency model in a fuzzy environment. evidences. Submitted to Int. Journal of Approximate Reasoning.

[8] Goldberg, D.E. (1989) Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, New York.

[9] Herrera, F., Lozano, M., and Verdegay, J.L. (1993) Tuning Fuzzy Logic Controllers by Genetic Algorithms, Dept. of Computer Science and Artificial Intelligence, Technical Report #DECSAI-93102.

[10] Herrera, F., Lozano, M., and Verdegay, J.L. (1993) Crossover Operators and Offspring Selection for Real Coded Genetic Algorithms, Dept. of Computer Science and Artificial Intelligence, Technical Report #DECSAI-93113.

[11] Lee, C.C. (1991) A self-learning rule-based controller employing approximate reasoning and neural net concepts, International Journal of Intelligent Systems 6 71-93.

[12] Michalewicz, Z. (1992) Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, New York.

[13] Nomura, H., Hayashi I., and Wakami, N. (1992) A Learning Method of Simplified Fuzzy Reasoning by Genetic Algorithm. Proc. of the Int. Fuzzy Systems and Intelligent Control Conference, 236-245, Lousville, KY.

[14] Pedricz, W. (1984) An identification algorithm in fuzzy relational systems. Fuzzy Sets and Systems 13 153-167.

[15] Peng, X., and Wang, P. (1988) On generating linguistic rules for fuzzy models, Lectures Notes in Computer Science 313, Uncertainty and Intelligent Systems (B. Bouchon, L. Saitta, R.R. Yager, Eds.), 185-192.

[16] Sugeno, M., and Kang, G.T. (1988) Structure identification of fuzzy model. Fuzzy Sets and Systems 28 15-33.

[17] Sugeno, M., and Tanaka, K. (1991) Successive identification of a fuzzy model and its applications to prediction of a complex system. Fuzzy Sets and Systems 42 315-334.

[18] Takagi, T., and Sugeno, M. (1985) Fuzzy Identification of Systems and Its Applications to Modeling and Control. IEEE Trans. Sys. Man and Cyber., Vol. SMC-15 No. 1, 116-132.

[19] Wang, L.X. (1992) Fuzzy Systems are Universal Approximators, IEEE Trans. Sys. Man and Cyber., Vol. SMC-7, No. 10, pp. 1163-1170.

[20] Yager, R.R., and Dimitar, P.F. (1993) Generation of Fuzzy Rules by Mountain Clustering. Iona College, TEch. Report #MII-1318.

[21] Yamaoka, M. and Mukaidono, M. (1991) A learning method of fuzzy inference rules with a neural network. Fourth IFSA Congress, Engineering 222-225.

[22] Yoshinari, Y., Pedrycz, W., and Hirota, K. (1993) Construction of fuzzy models through clustering techniques. Fuzzy Sets and Systems 54 157-165.