BIDIRECTIONAL INTEGRATION OF FUZZY LOGIC WITH GENETIC ALGORITHM AND LEARNING WITH GENETIC FUZZY SYSTEM

Riidhei Malhotra[1], Madhu Chauhan[2], , Uday Pratap Singh[3], and Mukul Pathak[4]

[1]Department of Information Technology, Galgotias College of Engineering & Technology, Greater Noida (U.P.), India

[2,3,4]Department of Computer Science & Engineering, Galgotias College of Engineering &Technology, Greater Noida(u.p), India

ABSTRACT

Recently, numerous papers and applications combining Fuzzy Logic (FL) and Genetic Algorithms (GAs) have become known, and there is an increasing interest in the integration of these two topics. In this paper we explore this combination from the bidirectional integration: The use of FL based techniques for both improving GA behavior and modeling GA components, the results obtained have been called fuzzy genetic algorithms (FGAs), and includes learning with genetic fuzzy systems i.e its different approaches. An analysis of genetic fuzzy rule based system and Genetic Tuning of Fuzzy Rule Based Systems including Basic Models.

KEYWORDS: GENETIC ALGORITHM, FUZZY LOGIC,LINGUISTIC VARIABLE, LEARNING WITH GENETIC ALGORITHM, GENETIC TUNING

1 INTRODUCTION

A Fuzzy Genetic Algorithm (FGA) is considered as a Genetic Algorithm (GA)that uses fuzzy logic based techniques or fuzzy tools to improve the GA behavior modeling different GA components. An FGA may be defined as an ordering sequence of instructions in which some of the instructions or algorithm components may be designed with fuzzy logic based tools, such as, fuzzy operators and fuzzy connectives for designing genetic operators with different properties, fuzzy logic control systems for controlling the GA parameters according to some performance measures, fuzzy stop criteria, representation tasks, etc.

1.1 GENETIC ALGORITHM (GA)

Genetic algorithms (GAs) have had a great measure of success in search and optimization problems.
The reason for a great part of their success is their ability to exploit the information accumulated about an initially unknown search space in order to bias subsequent searches into useful subspaces, i.e., *their adaptation*. This is their key feature, particularly in large, complex, and poorly understood search spaces, where classical search tools (enumerative, heuristic...) are inappropriate, offering a valid approach to problems requiring efficient and effective search techniques. GAs is general purpose search algorithms which use principles inspired by natural genetic populations to evolve solutions to problems [9]. The basic idea is to maintain a population of chromosomes, which represent candidate solutions to the concrete problem that evolves over time through a process of competition and controlled variation. Each chromosome in the population has an associated *fitness* to determine which chromosomes are used to form new ones in the competition process, which is called *selection*. The new ones are created using genetic operators such as *crossover* and *mutation*.
A GA starts off with a population of randomly generated *chromosomes*, and advances toward better *chromosomes* by applying genetic operators modeled on the genetic processes occurring in nature. The population undergoes evolution in a form of natural selection. During successive iterations, called *generations*, chromosomes in the population are rated for their adaptation as solutions, and on the basis of these evaluations, a new population of chromosomes is formed using a selection mechanism and specific genetic operators such as crossover and mutation. An *evaluation* or *fitness function* (f) must be devised for

each problem to be solved. Given a particular chromosome, a possible solution, the fitness function returns a single numerical fitness, which is supposed to be proportional to the utility or adaptation of the solution represented by that chromosome.
Although there are many possible variants of the basic GA, the fundamental underlying mechanism consists of three operations:
1. Evaluation of individual fitness,
2. Formation of a gene pool (intermediate population) through selection mechanism, and
3. Recombination through crossover and mutation operators.

1.2 FUZZY LOGIC (FL)

Fuzzy logic is a form of **many-valued logic**; it deals with reasoning that is approximate rather than fixed and exact. In contrast with traditional logic theory, where binary sets have two-valued logic: true or false, fuzzy logic variables may have a truth value that ranges in degree between 0 and 1. Fuzzy logic has been extended to handle the concept of partial truth, where the truth value may range between completely true and completely false. Furthermore, when linguistic variables are used, these degrees may be managed by specific functions.

Linguistic variables

While variables in mathematics usually take numerical values, in fuzzy logic applications, the non-numeric *linguistic variables* are often used to facilitate the expression of rules and facts. A linguistic variable such as *age* may have a value such as *young* or its antonym *old*. However, the great utility of linguistic variables is that they can be modified via linguistic hedges applied to primary terms. The linguistic hedges can be associated with certain functions.

1.3 LEARNING WITH GENETIC ALGORITHMS

Although GAs are not learning algorithms, they may offer a powerful and domain-independent search method for a variety of learning tasks. In fact, there has been a good deal of interest in using GAs for machine learning problems([7,10,8]).Three alternative approaches, in which GAs have been applied to learning processes, have been proposed, the Michigan , the Pittsburgh ([17]), and the Iterative Rule Learning (IRL) approaches [20]. In the first one, the chromosomes correspond to classifier rules which are evolved as a whole, whereas in the Pittsburgh approach, each chromosome encodes a complete set of classifiers. In the IRL approach each chromosome represents only one rule learning, but contrary to the first, only the best individual is considered as the solution, discarding the remaining chromosomes in the population. Below , we will describe them briefly.

Michigan Approach. The chromosomes are individual rules and a rule set is represented by the entire population. The collection of rules is modified over time via interaction with the environment. This model maintains the population of classifiers with credit assignment, rule discovery
and genetic operations applied at the level of the individual rule. A genetic learning process based on the Michigan approach receives the name of Classifier System. A complete description is to be found in [4].
Pittsburgh Approach. Each chromosome encodes a whole rule sets. Crossover serves to provide a new combination of rules and mutation provides new rules. In some cases, variable-length rule bases are used, employing modified genetic operators for dealing with these variable-length and position independent genomes. This model was initially proposed by Smith in 1980 [17]. Recent instances of this approach may be found in [10].
Iterative Rule Learning approach. In this latter model, as in the Michigan one, each chromosome in the population represents a single rule, but contrary to the Michigan one, only the best individual is considered to form part of the solution, discarding the remaining chromosomes in the population. Therefore, in the iterative model, the GA provides a partial solution to the problem of learning. In order to obtain a set of rules, which will be a true solution to the problem, the GA has to be placed within an iterative scheme similar to the following:
1. Use a GA to obtain a rule for the system.
2. Incorporate the rule into the final set of rules.
3. Penalize this rule.
4. If the set of rules obtained till now is adequate to be a solution to the problem, the system ends up returning the set of rules as the solution. Otherwise return to step 1.
The main difference with respect to the Michigan approach is that the fitness of each chromosome is computed individually, without taking into account cooperation with other ones. This substantially reduces the search space, because in each sequence of iterations only one rule is searched.

2. GENETIC FUZZY RULE BASED SYSTEMS (GFRBS)

The idea of a Genetic FRBS is that of a genetic FRBS design process which incorporates genetic techniques to achieve the automatic generation or modification of its KB (KNOWLEDGE BASE) (or a part of it). This generation or modification usually involves a tuning/learning process, and consequently this process plays a central role in GFSs. The objective of this tuning/learning process is optimization, i.e., maximizing or minimizing a certain function representing or describing the behavior of the system. It is possible to define two different groups of optimization problems in FRBSs. The first group contains those problems where optimization only involves the behavior of the FRBS, while the second one refers to those problems where optimization involves the global behavior of the FRBS and an additional system. The first group contains problems such as modeling, classification, prediction and, in general, identification problems. In this case, the optimization process searches for an FRBS able to reproduce the behavior of a certain target system. The most representative problem in the second group is control, where the objective is to add an FRBS to a controlled system in order to obtain a certain overall behavior. Next, we analyze some aspects of the Genetic FRBSs.

2.1 Obtaining the Knowledge for an FRBS

As a first step, it is interesting to distinguish between tuning and learning problems. In tuning problems, a predefined RB is used and the objective is to find a set of parameters defining the Database(DB).
In learning problems, a more elaborate process including the modification of the Rule Base(RB) is performed. We can distinguish between three different groups of GFSs depending on the KB components included in the genetic learning process.

Genetic tuning of the DB. The tuning of the scaling functions and fuzzy membership functions is an important task in the design of fuzzy systems. It is possible to parameterize the scaling functions or the membership functions and adapt them using GAs to deal with their parameters according to a fitness function. As regards to the tuning of membership functions, several methods have been proposed in order to define the DB using GAs. Each chromosome involved in the evolution process represents different DB definitions, i.e., each chromosome contains a coding of the whole set of membership functions giving meaning to the linguistic terms. Two possibilities can be considered depending on whether the fuzzy model nature is descriptive or approximate, either to code the fuzzy partition maintaining a linguistic description of the system, or to code the rule membership functions tuning the parameters of a label locally for every rule, thereby obtaining a fuzzy approximate model.

Genetic learning of the RB. All the methods belonging to this family involve the existence of a predefined collection of fuzzy membership functions giving meaning to the linguistic labels contained in the rules, a DB. On this basis GAs are applied to obtain a suitable rule base, using chromosomes that code single rules or complete rule bases.
Genetic learning of the KB. There are many approaches for the genetic learning of a complete KB (RB and DB). We may find approaches presenting variable chromosome lengths, others coding a fixed number of rules and their membership functions, several working with chromosomes encoding single control rules instead of a complete KBs, etc.

3. Genetic Tuning of Fuzzy Rule Based Systems: Basic Models

The tuning of the scaling functions and fuzzy membership functions is an important task in the design of fuzzy systems. It is possible to parameterize the scaling functions or the membership functions and adapt them using Genetic Algorithms to deal with their parameters according to a fitness function.
As regards to the tuning of membership functions, several methods have been proposed in order to define the Data Base (DB) using GAs. Each chromosome involved in the evolution process represents different DB definitions, i.e., each chromosome contains a coding of the whole set of membership functions giving meaning to the linguistic terms. Two possibilities can be considered depending on whether the fuzzy model nature is descriptive or approximate, either to code the fuzzy partition maintaining a linguistic description of the system, or to code the rule membership functions tuning the parameters of a label locally for every rule, thereby obtaining a fuzzy approximate model.
In this we analyze the use of GAs for the tuning of DBs according to the two mentioned areas, the adaptation of contexts using scaling functions and the tuning of membership functions, we shall present briefly them.

3.1 Adapting the Context

The use of scaling functions that are applied to the input and output variables of an FRBS, allows us to work with normalized universes of discourse where the fuzzy membership functions are defined. These scaling functions could be interpreted as gains associated with the variables (from a control engineering point of view) or as context information that translates relative semantics into absolute ones (from a knowledge engineering point of view). If using scaling functions, it is possible to fix them or to parameterize the scaling functions and adapt them. Linear and non-linear contexts have been used.
Linear context. It is the simplest scaling. The parameterized function is defined by means of two parameters (one, if used as a scaling factor). The

effect of scaling is that of linearly mapping the real interval [a,b] into a reference interval (e.g., [0,1]). The use of a scaling factor maps the interval [-a,a] in a symmetrical reference interval (e.g., [-1,1]). This kind of context is the most broadly applied one. Genetic techniques have been applied to adapting the parameters defining the scaling factors and linear scaling functions ([16]).

Nonlinear context. The main disadvantage of linear scaling is the fixed relative distribution of the membership functions (uniformly distributed or not) once they have been generated. To solve this problem nonlinear scaling is used allowing us to obtain a modified relative distribution and a change in the shape of the membership functions. The definition of parameterized nonlinear scaling functions is more complex than in the linear case and a larger number of parameters are needed. The process actually requires two steps: previous scaling (linear) and nonlinear mapping. Parameterized potential and sig modal ([11]) functions have been used when applying Gas to adapt the nonlinear context. Usually, the parameters (real numbers) constitute the genes of the chromosomes without binary representation.

Figure 3.1 shows a normalized fuzzy partition (top), a nonlinear adaptation with lower granularity for middle or for extreme values (center) and lower granularity for lowest or for highest
Values (bottom).

3.2 Tuning the Membership Functions

Another element of the KB is the set of membership functions. This is a second point where Gas could be applied with a tuning purpose. As in the previous case of scaling functions, the main idea is the definition of parameterized functions and the subsequent adaptation of parameters. Some approaches are found to be in [1, 12, 14]. The different proposals differ in the coding scheme and the management of the solutions (fitness functions,)

3.2.1 Shape of the Membership Functions

Two main groups of parameterized membership functions have been proposed and applied: piecewise linear functions and differentiable functions.

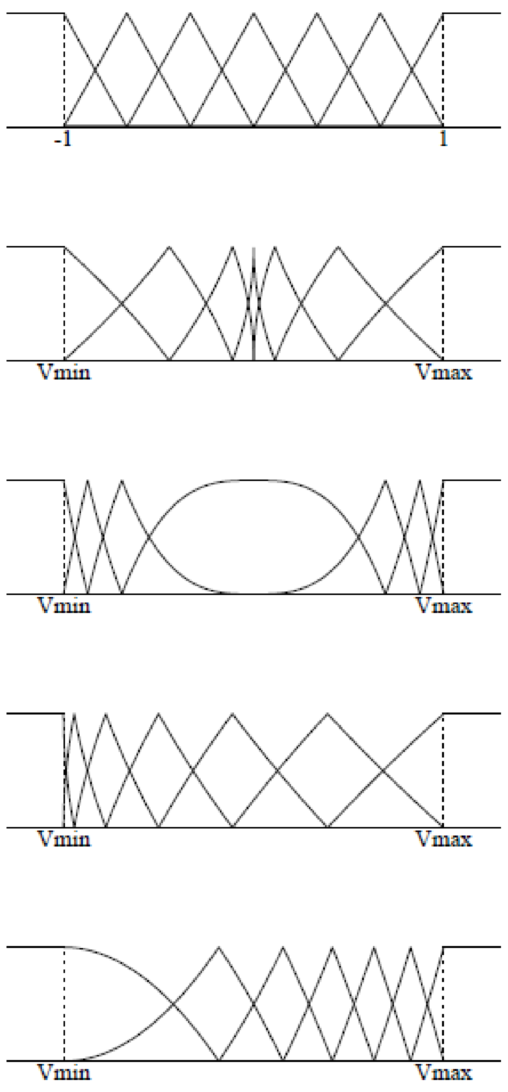
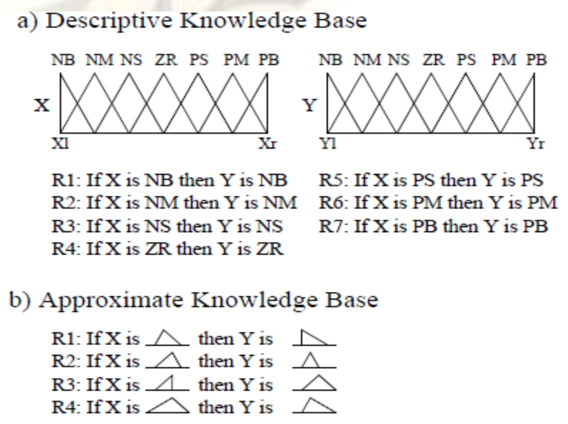


Fig 3.1: Non Linear Contexts Adaptation

Fig 3.2: Descriptive Versus Approximate Fuzzy Models



R1: If X is NB then Y is NB
R2: If X is NM then Y is NM
R3: If X is NS then Y is NS
R4: If X is ZR then Y is ZR
R5: If X is PS then Y is PS
R6: If X is PM then Y is PM
R7: If X is PB then Y is PB

Piecewise linear functions. The most broadly used parameterized membership functions in the field of GFSs are triangles, in some cases these are isosceles and other times they are irregular. A second possibility is trapezoidal membership functions. Each parameter of the function constitutes a gene of the chromosome that may be a binary code representing the parameter or a real number (the parameter itself,). Gaussian, bell and sigmoidal are examples of parameterized differentiable functions.

3.2. Scope of the Semantics

The genetic tuning process of membership functions is based on two variants, depending on the fuzzy model nature, whether approximate ([12]) or descriptive (5, 14]). The descriptive fuzzy model is essentially a qualitative expression of the system. A KB in which the fuzzy sets giving meaning (semantic) to the linguistic labels are uniformly defined for all rules included in the RB. It constitutes a descriptive approach since the linguistic labels take the same meaning for all the fuzzy rules contained in the RB. Concluding Remarks 27 In the approximate fuzzy model a KB is considered for which each fuzzy rule presents its own meaning, i. e., the linguistic variables involved in the rules do not take as their values any linguistic label from a global term set. In this case, the linguistic variables become fuzzy variables. The system applies local semantics. Figure 3.2 and the examples described in the following paragraphs illustrate these two variants, and their particular aspects reflected in the coding scheme.

3.2.3 The Approximate Genetic Tuning Process

As mentioned earlier, each chromosome forming the genetic population will encode a complete KB. More concretely, all of them encode the RB, R, and the difference between them is the fuzzy rule membership functions, i. e., the DB definition. Taking into account a parametric representation with triangular-shaped membership functions based on a 3-tuple of real values, each rule Ri : IF x1 is Ai1 and ... and xn is Ain THEN y is Bi, of a certain KB (KBl), is encoded in a piece of chromosome Cli: Cli = (ai1; bi1; ci1; : : : ; ain; bin; cin; ai; bi; ci) where Aij , Bi have the parametric representation (aij ; bij ; cij), (ai; bi; ci), i = 1; : : : ;m (m represents the number of rules), j = 1; : : : ; n (n is the number of input variables). Therefore the complete RB with its associated DB is represented by a complete chromosomeCl: Cl = Cl1 Cl2 ::: Clm .This chromosome may be a binary or a real coded individual.

3.2.4 The Descriptive Genetic Tuning Process

In this second genetic tuning process each chromosome encodes a different DB definition based on the fuzzy domain partitions. A primary fuzzy partition is represented as an array composed by 3 _ N real values, with N being the number of terms forming the linguistic variable term set. The complete DB for a problem, in which m linguistic variables are involved, is encoded into a fixed length real coded chromosome Cj built up by joining the partial representations of each one of the variable fuzzy partitions,

Cji = (ai1; bi1; ci1; : : : ; aiNi ; biNi ; ciNi)

Cj = Cj1 Cj2 ::: Cjm

where Cji represents the fuzzy partition corresponding to the i th variable.

4. Learning with Genetic Fuzzy Systems: Pittsburgh Approach

4.1 Introduction

Recently, there has been a growing interest in using Genetic Algorithms (GAs) for machine learning problems, appearing different genetic learning approaches. One of them, the Pittsburgh approach adopts the view that each individual in a population, each chromosome, encodes a whole rule sets. Crossover serves to provide a new combination of rules and mutation provides new rules. In some cases, variable-length rule bases are used, employing modified genetic operators for dealing with these variable-length and position independent genomes. This model was initially proposed by Smith in 1980 [17]. Here, we shortly describe the use of Genetic Fuzzy Systems (GFSs) with this learning approach for learning Rule Bases (RB) and Knowledge Bases (KB) for Fuzzy Rule Bases Systems (FRBSs).

4.2 Genetic Learning of RB

It is possible to represent the RB of an FRBS with three different representations. These representations are: relational matrix, decision table and list or set of rules. The Pittsburgh approach has been applied to learn rule bases in two different situations. The first situation refers to those systems using a complete rule base represented by means of a decision table or a relational matrix. The second situation is that of FRBSs, whose RB is represented using a list or set of fuzzy rules.

4.2.1 Using a Complete RB

A tabular representation guarantees the completeness of the knowledge of the FRBS in the sense that the coverage of the input space (the Cartesian product of universes of the input variables) is only related to the level of coverage of each input variable (the corresponding fuzzy partitions), and not to the rules.

Decision tables. A possible representation for the RB of an FS is a decision table. It is a classical representation used in different GFSs. A chromosome is obtained from the decision table by going row-wise and coding each output fuzzy set as an integer or any other kind of label. It is possible to include the "no output" definition in a certain position, using a "null" label ([18]).

Relational matrices. Occasionally GAs are used to modify the fuzzy relational matrix (R) of a Fuzzy System with one input and one output. The chromosome is obtained by concatenating the m _ n elements of R, where m and n are the number of fuzzy sets associated with the input and output variables respectively. The elements of R that will make up the genes may be represented by binary codes or real numbers.

4.2.2 Using a Partial RB

Neither the relational nor the tabular representations are adaptable to systems with more than two or three input variables because of the dimension of a complete RB for these situations. This fact stimulated the idea of working with sets of rules. In a *set of rules* representation the absence of applicable rules for a certain input that was perfectly covered by the fuzzy partitions of individual input variables is possible. As a counterpart to the loss of completeness, this representation allows *compressing* several rules with identical outputs into a singular rule and this is a really important question as the dimension of the system grows. There are many different methods for coding the rule base in this kind of evolutionary system. The code of the rule base is usually obtained by concatenating rules codes.

Rules of fixed length. A first approach is to represent a rule with a code of fixed length and position dependent meaning. The code will have as many elements as the number of variables in the system. A possible content of these elements is: a label pointing to a certain fuzzy set in the fuzzy partition of the variable or a binary string with a bit per fuzzy set in the fuzzy partition of the variable coding the presence or absence of the fuzzy set in the rule [15].

Rules of variable length. Codes with position independent meaning and based on pairs {*variable, membership function*} (the membership functions is described using a label) are used in .

4.3 Genetic Learning of KB

The simultaneous use as genetic material of the DB and the RB of an FRBS has produced different and interesting results. The most general approach is the use of a set of parameterized membership functions and a list of fuzzy rules that are jointly coded to generate a chromosome, then applying a Pittsburgh-type GA to evolve a population of such chromosomes. This kind of GFSs use chromosomes

Containing two sub-chromosomes that encode separately, but not independently, the DB and the RB. It is possible to maintain, at this point, the same division that was stated when talking about genetic learning of RBs with a Pittsburgh approach: learning complete rule bases or partial rule bases.

4.3.1 Using a Complete RB

In the rule base is represented as a fuzzy relation matrix (R), and the GA modifies R or the fuzzy membership functions (triangular) or both of them simultaneously, on a Fuzzy Logic Controller (FLC) with one input and one output variables. Each gene is a real number. When generating the optimal fuzzy relation matrix this real number corresponds to a fuzzy relation degree whose value is between 0 and 1. The genetic string is obtained by concatenating the m _ n real numbers that constitute R. When finding simultaneously the optimal rule base and the fuzzy membership functions, each chromosome allocates two sub-chromosomes: the genes of the rule base and the genes of the fuzzy membership functions. Both sub-chromosomes are treated as independent entities as far as crossover and mutation are concerned but as a single entity as far as reproduction is concerned. A slightly different approach is to use a TSK-type rule base, structuring its genetic code as if it came from a decision table. In this case, the contents of the code of a rule base is an ordered and complete list containing the consequents of all possible rules, where the antecedents are implicitly defined as a function of the position the consequent occupies in the list. The fuzzy membership functions constitute a first sub-chromosome while the coefficients of the consequents for a TSK fuzzy model constitute the second sub-chromosome. One gene is used to code each coefficient of a TSK-type, a single coefficient is considered for the output.

4.3.2 Using a Partial RB

Liska and Melsheimer use a rule base defined as a set of a fixed number of rules, and code each rule with integer numbers that define the membership function related with a certain input or output variable that is applied by the rule (membership functions for every variable are ordered). The systems use radial membership functions coded through two real numbers (two genes). The genetic string is obtained by concatenating the two genes in each membership function. There are many different methods for coding the rule base in this kind of evolutionary system. The code of the rule base is usually obtained by concatenating rule codes. To represent a single rule, it is possible to use a position dependent code with as many elements as the number of variables of the system. A possible content in these elements is: a label pointing to a certain fuzzy set in the fuzzy partition of the variable or a binary string with a bit per fuzzy set in the fuzzy partition of the variable. Using an approximate approach, include the definition of the membership functions into the rules, coding each rule through the corresponding set of membership functions.

5. Learning with Genetic Fuzzy Systems: Iterative Rule Learning Approach

5.1 Introduction

Since the beginning of the 80s there has been growing interest in applying methods based on Genetic Algorithms (GAs) to automatic learning problems, especially the learning of production rules on the basis of attribute-evaluated example sets. The main problem in these applications consists of finding a "comfortable" representation in the sense that it might be capable both of gathering the problem’s characteristics and representing the potential solutions. In recent literature we may find different algorithms that use a new learning model based on GAs, the *Iterative Rule Learning (IRL) approach* [20]. In the latter model, as in the Michigan one, each chromosome in the population represents a single rule, but contrary to the latter, only the best individual is considered as the solution, discarding the remaining chromosomes in the population. This model has been used in papers such as [20, 13].

5.2 IRL Approach

In this approach the GA provides a partial solution to the problem of learning, and attempts to reduce the search space for the possible solutions. In order to obtain a set of rules, which will be a true solution to the problem, the GA has to be placed within an iterative scheme similar to the following:

1. Use a GA to obtain a rule for the system.
2. Incorporate the rule into the final set of rules.
3. Penalize this rule.
4. If the set of rules obtained is adequate to represent the examples in the training set, the system ends up returning the set of rules as the solution. Otherwise return to step 1.

A very easy way to penalize the rules already obtained, and thus be able to learn new rules, consists of eliminating from the training set all those examples that are covered by the set of rules obtained previously. This learning way is to allow "niches" and "species" formation. Species formation seems particularly appealing for concept learning, considering the process as the learning of multimodal Concepts. The main difference with respect to the Michigan approach is that the fitness of each chromosome is computed individually, without taking into account cooperation with other ones. This Reduces substantially the search space, because in each sequence of iterations only one rule is searched.

In the literature we can find some genetic learning processes that use this model such as *SLAVE*, *SIAVE* [20] and the *genetic generation process*. These three genetic learning processes use the IRL approach with light difference: *SLAVE* launches a new GA to find a new rule after having eliminated the examples covered by the last rule obtained. SLAVE was designed to work with or without linguistic information .*SIAVE* uses a single GA that goes on detecting rules and eliminating the examples covered by the latter. SIA can only work with crisp data. The *genetic generation process* runs a GA for obtaining the best rule according to different features, assigns a relative covering value to every example, and removes the examples with a covering value greater than a constant.

From the description above, we may see that in order to implement learning algorithm based on GAs using the IRL approach, we need, at least, the following:

1. A criterion for selecting the best rule in each iteration,
2. A penalty criterion, and
3. A criterion for determining when enough rules are available to represent the examples in the training set.

The first criterion is normally associated with one or several characteristics that are desirable so as to determine good rules. Usually criteria about the rule strength have been proposed (number of examples covered), criteria of consistency of the rule or criteria of simplicity.

The second criterion is often associated, although it is not necessary, with the elimination of the examples covered by the previous rules.

Finally, the third criterion is associated with the completeness of the set of rules and must be taken into account when we can say that all the examples in the training set are sufficiently covered and no more rules are needed to represent them.

5.2.1 Multi-Stage Genetic Fuzzy System Based on the IRL Approach

Learning algorithms that use the IRL approach do not envisage any relationship between them in the process for obtaining rules. Therefore, the final set of rules usually needs an a posteriori process that will modify and/or fit the said set. The methodology that is presently applied includes different processes that are not necessarily applied simultaneously. This methodology, which we call *multi-stage genetic fuzzy*

systems and has been abbreviated as MSGFS, consists of three component parts:
I A *genetic generation stage* for generating fuzzy rules using the IRL approach.
II A *post-processing stage* working on the rule set obtained in the previous stage in order to either to refine rules or eliminate redundant rules.
III A *genetic tuning stage* that tunes the membership functions of the fuzzy rules.
We describe these shortly below.

Genetic generation stage. In this stage the IRL approach is used for learning fuzzy rules capable of including the complete knowledge from the set of examples. A chromosome represents a fuzzy rule, the generation method selects the best rule according to different features included in the fitness function of the GA, features that include general properties of the KB and particular requirements to the fuzzy rule. This features lead to the definition of the covering degree between a rule and an example and the use of the concept of positive and negative examples. The IRL approach uses a covering method of the set of examples. This covering method assigns a relative covering value to every example, and removes the examples with an adequate covering value, according to a covering criterion. As we have indicated, this model may be used for learning RB as *SLAVE* and for learning KB as the *genetic generation process* proposed in [13].
Post-processing stage: selection and refinement. As we mentioned earlier, the IRL approach does not analyze any relationship between the rules that it is obtaining. That is why, once the rule base has been obtained, it may be improved either because there are rules that may be refined or redundant rules if high degrees of coverage are used. Two possible post-processing methods have been used , a refinement algorithm and a selection or simplification algorithm [12].
Genetic tuning stage. At this stage *the genetic tuning process* is applied over the KB for obtaining a more accurate one. We can consider two possibilities, depending on the fuzzy model's nature:
a) an approximate model based on a KB composed of a collection of fuzzy rules without a fixed relationship between the fuzzy rules and some primary fuzzy partitions giving meaning to them, or
b) a descriptive model based on a linguistic description of the system with a fuzzy partition that assigns a membership function to every linguistic label. In both cases, each chromosome forming the genetic population will encode a complete DB, but in the first case each piece of chromosome codes the membership functions associated to one rule and in the second one each piece of chromosome codes the fuzzy partition of a variable. The main difference between both processes is the coding scheme.

5.2.2 A Multi-stage Genetic Fuzzy Rule-Based System Structure

In the following we present a guideline structure for multi-stage GFRBSs used in [13]:

a) A Fuzzy Rule Generation Process. This process will determine the type of the final FRBS generated, so the generated fuzzy rules may present a descriptive, constrained approximate or unconstrained approximate semantics. In all cases, it will present two components: a *fuzzy rule generating method* composed of an inductive or evolutionary process which uses a niche criterion for obtaining the best possible cooperation among the fuzzy rules generated when working with the approximate approach, and an *iterative covering method* of the system behavior example set, which penalizes each rule generated by the fuzzy rule generating method by considering its covering over the examples in the training set and removes the ones yet covered from it. This process allows us to obtain a set of fuzzy rules with a concrete semantics covering the training set in an adequate form.

b) A Genetic Multi-Simplification Process for selecting rules, based on a binary coded GA with a phenotypic sharing function and a measure of the FRBS accuracy in the problem being solved. It will save the overlearning that the previous component may cause due to the existence of redundant rules, with the aim of obtaining a simplified KB presenting the best possible cooperation among the fuzzy rules composing it. This process will obtain different possibilities for this simplified KB thanks to a genotypic niching scheme.

c) An Evolutionary Tuning Process based on any kind of real coded EA and a measure of the FRBS performance. It will give the final KB as output by adjusting the membership functions for each fuzzy rule in each possible KB obtained from the genetic multi-simplification process. The type of tuning performed will depend on the nature of the FRBS being generated, i.e., when generating a descriptive FRBS, a global tuning of the fuzzy partition associated to each linguistic variable will be performed, but when working with any of the approximate approaches, the membership functions involved in each fuzzy rule will be adjusted. The most accurate KB obtained in this stage will constitute the final output of the whole learning process

6. **Learning with Genetic fuzzy system: Michigan Approach**

6.1 INTRODUCTION

While classifier systems of the Michigan type had been introduced by J. H. Holland in 1976, their fuzzification awaited discovery many years. The first fuzzy classifier system of the Michigan type was introduced by M. Valenzuela-Rendón ([19]) and is, more or less, a straightforward fuzzification of a Holland classifier system. An alternative approach has been developed by A. Bonarini ([2, 3]), who applies a different scheme of competetion between classifiers. These two approaches have in common that they operate only on the rules — the shape of the membership functions is fixed. A third method, which was introduced by P. Bonelli and A. Parodi, tries to optimize even the membership functions and the output weights in accordance to payoff from the environment.

6.2 Fuzzifying Holland Classifier Systems

6.2.1 The Production System

We consider a fuzzy controller with real-valued input and output. The system has, unlike ordinary fuzzy controllers, three different types of variables — input, output, and internal variables. As we will see later, internal variables are for the purpose of storing information about the near past. They correspond to the internally tagged messages in Holland classifier systems. For the sake of generality and simplicity, all the universes of discourse, are transformed to the unit interval [0; 1]. For each variable the same number of membership functions n is assumed. These membership functions are fixed at the beginning. They are not changed throughout the learning process. M. Valenzuela-Rendón took bell-shaped function which divided the interval rather equally. A message is a binary string of length l + n, where n is the number of membership functions defined above and l is the length of the prefix (tag), which identifies the variable to which the message belongs. A good choice for l would be dlog2 Ke, where K is the total number of variables we want to consider. To each message an *activity level*, which represents a truth value, is assigned.
Consider for instance the following message (l = 3, n = 5): |0{1z0}=2: 00010 ! 0:6
Its meaning is “Input no. 2 belongs to fuzzy set no. 4 with a degree of 0:6”. On the message list only so-called minimal messages are used, i.e., messages with only one 1 in the part which identifies the numbers of the fuzzy sets. Classifiers again consist of a fixed number r of conditions and an action part. Note that, in this approach, no wildcards and no “–” prefixes are used. Both condition and action part are also binary strings of length l +n, where the tag and the identifiers of the fuzzy sets are separated by a colon. Then the degree to which such a condition is matched is a truth value between 0 and 1. The degree of matching is computed as the maximal activity of messages on the list, which have the same tag and whose 1s are a subset of those of the condition. Figure 6.1 shows a simple example how this matching is done. The degree of satisfaction of the whole classifier is then computed as the minimum of matching degrees of the conditions. This is then also the activity level which is assigned to the output message (i.e., Mamdani inference).

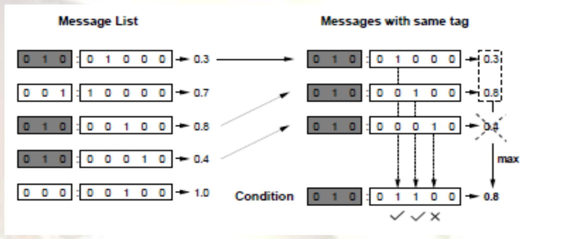


Fig6.1: Matching a fuzzy condition

The whole rule base consists of a fixed number m of such classifiers. Similarly to Holland classifier systems, one execution step of the production system is done as follows:
1. The detectors receive crisp input values from the environment and translate them into minimal messages which are then added to the message list.
2. The degrees of matching are computed for all classifiers.
3. The message list is erased.
4. The output messages of some matched classifiers are placed on the message list.
5. The output messages are translated into minimal messages. For instance, the message 010 : 00110 ! 0:9 is split into the two messages 010 : 00010 ! 0:9 and 010 : 00100 ! 0:9.
6. The effectors discard the output messages (referring to output variables) from the list and translate them into instructions to the environment.
From point 2 it can be seen easily that it is of advantage to use fuzzy sets with local support instead of bell-shaped ones, because, if bell-shaped fuzzy sets are used, every rule fires in each time step.
Step 6 is done by a modified Mamdani inference: The sum (instead of the maximum or another t-conorm) of activity levels of messages, which refer to the same fuzzy set of a variable, is computed. The membership functions are then scaled with these sums. Finally, the center of gravity of the “union” (i.e. maximum) of these functions, which belong to one variable, is computed (Sum-Prod inference).

6.2.2 Rule Discovery

The adaptation of a genetic algorithm to the problem of manipulating classifiers in our system is again straightforward. We only have to take special care that tags in conditional parts must not refer to output variables and that tags in the action parts of the classifiers must not refer to input variables of the system. Analogously to our previous considerations, if we admit a certain number of internal variables, the system tends to build up internal chains, coupled sequences, autonomously. If we admit internal variables, a classifier system of this type not only learns stupid input-output actions, it also tries to discover causal interrelations.

6.3 Bonarini's ELF Method

In [2], A. Bonarini presents his ELF (=evolutionary learning of fuzzy rules) method and applies it to the problem of guiding an autonomous robot. The key issue of ELF is to find a small rule base which only contains important rules. While he takes over many of M. Valenzuela-Rendón's ideas, his way of modifying the rule base differs strongly from Valenzuela-Rendón's straightforward fuzzification of Holland's technique. Bonarini calls the modification scheme "cover-detector algorithm". The number of rules can be varied in each time step depending on the number of rules which match the actual situation.
This is done by two mutually exclusive operations:
1. If the rules, which match the actual situation, are too many, the worst of them is deleted.
2. If there are too few rules matching the current inputs, a new rule, whose antecedents cover the current state, with randomly chosen consequent value, is added to the rule base.
The genetic operations are only applied to the consequent values of the rules. Since the antecedents are generated on demand in the different time steps, no taxation is necessary.
Seemingly, such a simple modification scheme can only be applied to so-called one-stage problems, where the effect of each rule can be observed in the next time step. For applications where this is not valid, e.g., backing up a truck, Bonarini introduced a modification of his ELF algorithm — the concept of an *episode*, which is a given number of subsequent control actions, after which they reached state is evaluated.

6.4 Online Modification of the Whole Knowledge Base

While the last two methods only manipulate rules and work with fixed membership functions, there is at least one variant of fuzzy classifier systems were also the membership functions are involved in the learning process. This variant was introduced by A. Parodi and P. Bonelli in [65]. The main idea is that an approximate knowledge base is used instead of a descriptive one as in the two previous examples. So, a fuzzy rule is not represented as a linguistic expression which refers only to labels of fuzzy sets, but a fuzzy relation on X_Y , where X is the input and Y is the output domain. More specifically, each rule is represented as a pair consisting of a fuzzy subset of X and a fuzzy subset of Y .Since, in many applications, X and Y are themselves cross products, i.e., X = X1__ _ __Xn and Y = Y1 _ _ _ _ _ Ym, rules in a approximative knowledge base can be written as Ai1 _ _ _ _ _ Ain _ Bi1 _ _ _ _ _ Bim . Where i is the index of the rule. If one restricts to certain class of fuzzy subsets, such as triangular or bell-shaped membership functions, it is possible to encode a rule as (ai1; : : : ; ain; bi1; : : : ; bim)
where aij and bij are parameters uniquely identifying a fuzzy subset of Xj or Yj , respectively. Moreover, in this approach, each rule is additionally equipped with a strength factor, which is taken as a scaling factor of the output set. This strength factor is also used as fitness measure by the genetic algorithm which modifies the knowledge base and modified according to payoff from the environment.

7. Conclusion

One of the most important advantages of fuzzy systems is that the functions are parameterized in a way which is interpretable for humans. More specifically, it is possible to translate human knowledge into fuzzy rules and fuzzy sets, but, on the contrary, not every system, which is formally a fuzzy system, is really interpretable. In fact, the probability, that difficultly interpretable configurations are obtained, is rather high when representations with lots of degrees of freedom are tried to be optimized. An alternative, which can help to overcome this problem, is to encode whole fuzzy partitions as shown in the fifth lecture. Obviously, this approach allows less degree of freedom, which can also speed up convergence.
There have been a lot of publications concerning with genetic optimization of fuzzy systems (see [6] for recent bibliographies). Each of these approaches — many of them are rather similar—has only been applied to a few benchmark problems. So far, there are no proofs (neither theoretical nor empirical) which methods are suitable for which problems.

8.REFERENCES

[1] F. Bolata and A. Nowé. From fuzzy linguistic specifications to fuzzy controllers using evolution strategies. In *Proc. FUZZ-IEEE'95*, volume III, pages 1089–1094, 1995.

[2] A. Bonarini. ELF: Learning incomplete fuzzy rule sets for an autonomous robot. In *Proc. EUFIT'93*, volume I, pages 69–75, 1993.

[3] A. Bonarini. Evolutionary learning of fuzzy rules: Competition and cooperation. In W. Pedrycz, editor, *Fuzzy Modeling: Paradigms and Practice*, pages 265–283. Kluwer Academic
Publishers, Dordrecht, 1996.

[4] L. B. Booker, D. E. Goldberg, and J. H. Holland. Classifier systems and genetic algorithms.*Artificial Intelligence*, 40:235–282, 1989.

[5] O. Cordón, F. Herrera, and M. Lozano. A three-stage method for designing genetic fuzzy
systems by learning from examples. In H. M. Voight, W. Ebeling, I. Rechenberg, and H. P.
Schwefel, editors, *Proc. Int. Conf. on Parallel Problem Solving from Nature*, pages 720–729,
Berlin, 1994.

[6] O. Cordón, F. Herrera, and M. Lozano. A classified review on the combination
fuzzy logic–genetic algorithms. Technical Report DECSAI-95129, Dept. of Computer
Science and AI, University of Granada, Spain, December 1995. Available at
http://decsai.ugr.es/ herrera/fl-ga.html

[7] K. A. De Jong. Learning with genetic algorithms: An overview. *Mach. Learn.*, 3:121–138,1988.

[8] A. Giordana and F. Neri. Genetic algorithms in machine learning. *AI Communications*, 9:21–26, 1994.

[9] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*.
Addison-Wesley, Reading, MA, 1989.

[10] J. J. Grefenstette, editor. *Genetic Algorithms for Machine Learning*. Kluwer Academic Publishers, Boston, 1995.

[11] R. R. Gudwin, F. Gomide, and W. Pedrycz. Nonlinear context adaptation with genetic algorithms. In *Proc. IFSA'97*, 1997.

[12] F. Herrera, M. Lozano, and J. L. Verdegay. Tuning fuzzy logic controllers by genetic algorithms. *Internat. J. Approx. Reason.*, 12:299–315, 1995.

[13] F. Herrera, M. Lozano, and J. L. Verdegay. A learning process for fuzzy control rules using
genetic algorithms. *Fuzzy Sets and Systems*, 1997. to appear???????????

[14] C. L. Karr. Genetic algorithms for fuzzy controllers. *AI Expert*, 6(2):26–33, 1991.

[15] L. Magdalena. Adapting the gain of an FLC with genetic algorithms. *Internat. J. Approx.
Reason.*, 17(4):327–350, 1997.

[16] L. Magdalena and F. Monasterio. Evolutionary-based learning applied to fuzzy controllers. In *Proc. FUZZ-IEEE'95*, volume III, pages 1111–1118, 1995

[17] S. F. Smith. *A Learning System Based on Genetic Adaptive Algorithms*. PhD thesis, Universityof Pittsburgh, 1980.

[18] P. Thrift. Fuzzy logic synthesis with genetic algorithms. In R. K. Belew and L. B. Booker, editors, *Proc. ICGA'91*, pages 509–513, Los Altos, CA, 1991. Morgan Kaufmann.

[20] M. Valenzuela-Rendón. The fuzzy classifier system: A classifier system for continuously varying variables. In R. K. Belew and L. B. Booker, editors, *Proc. ICGA'91*, pages 346–353,
San Mateo, CA, 1991. Morgan Kaufmann.

[19] G. Venturini. SIA: A supervised inductive algorithm with genetic search for learning
attribute-based concepts. In *Proc. European Conf. on Machine Learning*, pages 280–296, 1993.