SLAVE: A genetic learning system based on an iterative approach *

Antonio GONZALEZ and Raúl PEREZ
Departamento de Ciencias de la Computación e Inteligencia Artificial
E.T.S. de Ingeniería Informática, Universidad de Granada, 18071-Granada (Spain)
E-mail: {A.Gonzalez,fgr}@decsai.ugr.es
Telefono: + 34.58.243199
Fax: + 34.58.243317

Abstract

SLAVE (Structural Learning Algorithm in Vague Environment) is an inductive learning algorithm that uses concepts based on fuzzy logic theory. This theory has shown be an useful representation tool for improving the understanding under a human point of view, of the knowledge obtained. Furthermore, SLAVE uses an iterative approach for learning with genetic algorithms. This method is an alternative approach from the classical Pittsburgh and Michigan approaches.

In this work, we propose some modifications of the original SLAVE learning algorithm, including new genetic operators for reducing the time needed for learning and improving the understanding of the rules obtained. Furthermore, we propose a new way for penalizing the rules in the iterative approach that permits to improve the behaviour of the system.

Keywords: machine learning, fuzzy logic, genetic algorithms.

1 Introduction

Inductive learning tries to extract a knowledge base that permits to describe the behaviour of a system from an example set that shows the behaviour of the system in the past. Thus, the automation of this kind of learning can be considered as an alternative to the classical techniques of knowledge acquisition [27]. Usually, this knowledge is represented through rules that represent the relationships between the different variables in the problem. Sometimes, some of these variables may be better represented as a linguistic variable [35], and in this case, we are using fuzzy rules to represent the knowledge of the system, and therefore we are trying to learn a system's behaviour through fuzzy models. In [17] the use of fuzzy domains was interpreted as a discretization technique. Several models for describing a system behaviour through the generation of fuzzy rules have been proposed in the literature [4, 8, 24, 26, 32].

Genetic Algorithms (GAs) are search algorithms that use operations found in natural genetic to guide the trek through a search space. GAs have been theoretically and empirically proven to provide robust search capabilities in complex spaces, offering a valid approach to problems requiring efficient and effective searching. An interesting application of GA has been the machine learning problems.

Two alternative approaches, in which GAs have been applied to learning processes are the Michigan [20] and the Pittsburgh [31] approaches. 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

*This work has been supported by CICYT under Project TIC95-0453

classifiers. A third way has been presented as an alternative to these models, the iterative approach [19] where each chromosomes represent an unique rule.

The iterative approach attempts to reduce the search space for the possible solutions. In this approach, 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. 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 where its mission consists of finding in each step a rule for the system. The obtained rule is incorporated into the final set of rules and it is penalized for permitting obtain new rules. This iterative scheme is repeated until the set of rules obtained is adequate to represent the examples in the training set, returning the set of rules as the solution of the problem.

Some learning algorithms use fuzzy models for representing the obtained knowledge [1, 2, 28, 32, 34]. The fuzzy models present two main advantage: they permit to work with imprecise data, providing a confortable way for their representation. In this sense, the missing values are representing in a natural way. In this second sense, the adquired knowlegde with these model may be closer to the knowlegde that the humans can understand.

SLAVE (Structural Learning Algorithm in Vague Environment) is an inductive learning algorithm based on the iterative approach and on fuzzy models that was initially proposed in [14] and later developed in [16, 18]. SLAVE begins with a simple description of the problem: the consequent variable (concept variable) and the set of all the available antecedent variables for generating the rules that describe the consequent variable. The learning algorithm, using this description and a set of examples, will decide for each class and each rule which are the variables needed to describe the concept (feature selection), and the rest will be eliminated.

In this work, we propose some modifications of the original SLAVE learning algorithm in the following two senses: first, we include new genetic operators for increasing the compresibility of the rules obtained and reducing the time and the rules needed for learning the concept. Second, we propose an alternative way for penalizing the rules in the iterative approach. The most common 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 enough covered by the set of rules obtained previously. This way of penalization rules, used in the original SLAVE, can cause some problems when we work with fuzzy rules and examples. With this methodology the information that has the system in order to obtain the next rule is the examples that were not covered by the previous rules from its class. The process for obtaining a new rule has not information about the degree in which the previous rules covered the eliminated examples from their classes and how these rules affect to the examples from the other classes. Therefore, the rule has no information over the behaviour of the rest of the rules. This problem, can cause unexpected interactions between the obtained rules in the inference process, that are not detected during the learning task. In order to avoid this problem, we include new information in the process of selecting a new rule by proposing a new interpretation of the positive and negative example concepts.

The next section is devoted to show the genetic iterative approach in opposite of the classical approaches. In section 3 the model of the rule used by SLAVE is presented. The initial SLAVE learning algorithm is described in section 4. Section 5 describes the modified version of SLAVE, and finally the last section shows a comparative study of the behaviour of the new learning algorithm on different databases with respect to the previous version and other learning algorithms.

2 The Iterative Model

GAs have shown themselves to be a very useful tool for the construction of learning algorithms [6, 7, 13]. GAs are search algorithms that use operations found in natural genetic to guide the trek through a search space. They are theoretically and empirically proven to provide robust search capabilities in complex spaces,

offering a valid approach to problems requiring efficient and effective searching.

Since the beginning of the 80s there has been growing interest in applying methods based on GAs to automatic learning problems, especially the learning of production rules on the basis of attributed-evaluated examples 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 potencial solutions.

Classically, two genetic learning approaches have been proposed:

- **The Michigan approach:** The members of the population are individual rules and a rule set is represented by the entire population. The collection of rules are modified over time via interaction with the environment. This model maintains the population of classifiers with credit assignament, rule discovery and genetic operations applied at the level of the individual rule.

  The first ideas appear in [20] and in [21] was developed the first classifiers system. From this moment, in the literature has been proposed different systems [9, 23, 28, 32].

- **The Pittsburgh approach:** Each chromosome encodes a whole classifier set. Credit is assigned to the complete set of rules via interaction with the enviroment. Crossover serves to provide a new combination of rules and mutation provides new rules. In some cases, variable-length classifier sets are used, employing modified genetic operators for dealing with these variable-length and position independent genomes.

  This model was initially proposed in [31]. Recent instances of this approach are the GABIL [7] and GIL [25].

An interesting discussion about the problems generated by the use of these approaches can be seen in [19]. In the recent literature we may find different algorithms that use a new learning model based on GAs called the iterative rule learning approach. In this approach, the GA provides a partial solution to the problem of learning. This model has been used in papers such as [3, 4, 14, 33] and attempts to reduce the search space for the possible solutions.

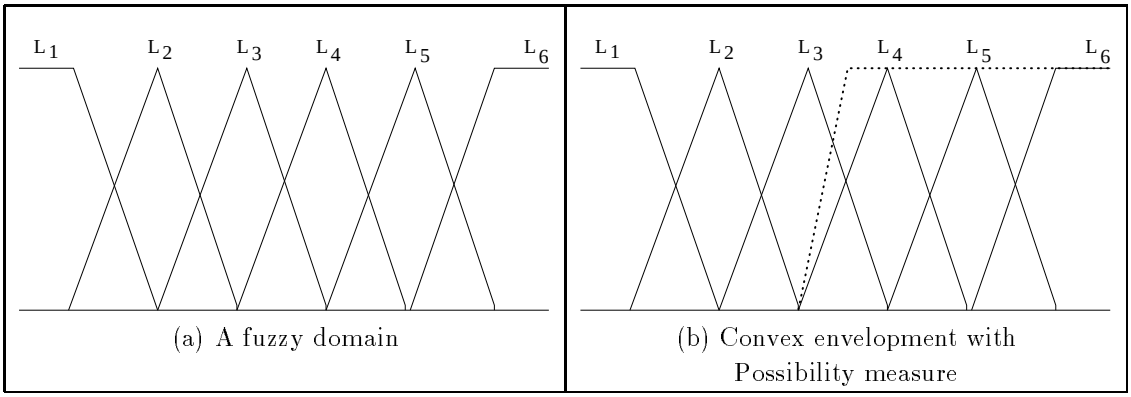
In order to obtain a set of rules, which will be a true solution to the problem, the GA has to be placed within an iterative scheme similar to the following [19]:

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 suffice 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.

This approach obtains in each step a rule, and the iteration of the process obtain the complete set of rules.

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 way of penalizing rules is used by some learning algorithm such as those in the AQ family [27] or the CN2 algorithm [5], and genetic learning algorithms such as [4, 16, 33].

When this model of penalization of rule is used in an iterative approach, can appear unexpected interactions between rules. These interactions can appear when an example is covered by several rules of different classes. We can solve this problem, as we will see later, through the addition of global information in each step of the iterative approach. This information consists of the best degree in which each example was covered by the rules of its class and the best degree in which each example was covered by the rules of rest of the classes.



(a) A fuzzy domain

(b) Convex envelopment with Possibility measure

Figure 1: Fuzzy domain

3 The rule model in SLAVE

The basic element of the SLAVE learning algorithm is its model of rules

$$\text{IF } X_1 \textit{ is } A_1 \textit{ and } \ldots \textit{ and } X_p \textit{ is } A_p \text{ THEN } Y \textit{ is } B$$

where each variable X_i has a referential set U_i and takes values in a finite domain D_i, for $i \in \{1, \ldots, p\}$. The referential set for Y is V and its domain is F. The value of the variable y is B, where $B \in F$ and the value of the variable X_i is A_i, where $A_i \in P(D_i)$ and $P(D_i)$ denotes the set of subsets of D_i.

In this model, we can consider, in general, that the variables are linguistics, i.e., the domains of the variable can be described using linguistic labels, or in general fuzzy sets.

The key to this rule model is that each variable can take as a value an element or a subset of element from its domain, i.e., we let the value of a variable be interpreted more as a disjunction of elements than just one element in its domain. This concept may be clarified with the following example

Example 1 *Let X_i be a variable which domain is shown in Figure 1(a). An antecedent like*

$$\ldots \textit{ and } X_i = \{L_4, L_5, L_6\} \textit{ and } \ldots$$

is equivalent to

$$\ldots \textit{ and } \{X_i \textit{ is } L_4 \textit{ or } X_i \textit{ is } L_5 \textit{ or } X_i \textit{ is } L_6\} \textit{ and } \ldots$$

Using the previous model of rule, the set of all possible rules is

$$\Delta = P(D_1) \times P(D_2) \times \ldots \times P(D_n) \times F.$$

In order to use this model of rule, in [16] we proposed the following concepts:

Definition 3.1 *Let a and b be two fuzzy sets in a common referential set U, and * a t-norm, we define the compatibility between a and b as the following function:*

$$\sigma(a, b) = \sup_{x \in U} \{\mu_a(x) * \mu_b(x)\}$$

where μ_c is the membership function of the c fuzzy set

Definition 3.2 *Let Dom_1 and Dom_2 be two domains made up by fuzzy sets in a common referential set U, and $C_1 \subseteq Dom_1$ and $C_2 \subseteq Dom_2$ each be a set of fuzzy sets. We define the compatibility between both sets as:*

$$\sigma(C_1, C_2) = \sup_{a \in C_1} \sup_{b \in C_2} \sigma(a, b)$$

We use the same symbol σ for both definitions since the last definition is a extension of the former, and both coincide for unitary sets.

From these concepts, in order to calculate the adaptation between a value v (crisp or fuzzy) and a set of fuzzy sets $L \subseteq D$, both over a X variable with fuzzy domain D, we use the following possibility measure

$$Poss(L|v) = \frac{\sigma(v, L)}{\sigma(v, D)}.$$

The value assigned to X_i in example 1 was $\{L_4, L_5, L_6\}$. Using the previous possibility measure, this value can be interpreted as "X_i is greater or iqual than L_4" since $Poss(\{L_4, L_5, L_6\}|v)$ is the normalized convex hull described in Figure 1(b).

Let e be an example composed by different features

$$e = (e_1, e_2, \ldots, e_n, c(e))$$

where e_i is now a fuzzy set in the referential set U_i, and $c(e)$ is the class assigned to the example e.

Usually, the features of the examples are crisp more than fuzzy, but without difficulty we can extend the kind of examples that the learning algorithm can use. In general to allow fuzzy examples is useful in order to include some possible imprecision in the data. For example, fuzzy inputs can appear when we have imprecise rules in which we have no complete confidence and we can decide include them as new examples to be considered. Moreover, fuzzy inputs can also appear when we have missing values of the kind "don't care" and we decide to replace them by its complete domain (a set of fuzzy sets) representing that any value is possible. In the previous examplet, the "don't care" value is replaced by $v = D$ and

$$Poss(L|D) = \frac{\sigma(D, L)}{\sigma(D, D)} = 1 \ \forall L \subseteq D$$

noindent when we work with normalized fuzzy labels.

Let $R_B(A)$ be a rule with antecedent part $A = (A_1, \ldots, A_n)$ $A_i \subseteq D_i$ and consequent part $B \in F$. We can define the adaptation between an example and the antecedent of a rule combining through a t-norm the measure of possibility of each A_i given the evidence supported by the example:

$$U(e, A) = Poss(A_1|e_1) * Poss(A_2|e_2) * \ldots * Poss(A_n|e_n).$$

In the same way, we define the adaptation between the example and the consequent of the rule as

$$U(e, B) = \frac{\sigma(c(e), B)}{\sigma(c(e), F)}.$$

From these concepts we can define the positive and negative example set given a rule.

Definition 3.3 *The set of positive examples for the rule $R_B(A)$ are the following fuzzy subsets of E*

$$E^+(R_B(A)) = \{(e, U(e, A) * U(e, B)) \mid e \in E\}$$

*where $A = (A_1, \ldots, A_n)$, and * is a t-norm.*

Definition 3.4 *The set of negative examples for the rule $R_B(A)$ are the following fuzzy subsets of E*

$$E^-(R_B(A)) = \{(e, U(e, A) * U(e, \overline{B})) \mid e \in E\}$$

*where $A = (A_1, \ldots, A_n))$, $\overline{B} = \bigcup_{b \in F}^{b \neq B} b$ and * is a t-norm.*

Definition 3.5 *The set of covered examples for the rule $R_B(A)$ are the following fuzzy subsets of E*

$$E(R_B(A)) = \{(e, U(e, A)) \mid e \in E\}$$

*where $A = (A_1, \ldots, A_n))$, and * is a t-norm.*

Once we have definitions for positive and negative examples of a rule, we can use a counter to determine the number of positive and negative examples for a rule. Thus, let us denote $|\phi| = \sum_{u \in U} \mu_\phi(u)$, as the cardinality of a fuzzy subset. By using this, we obtain the following definitions:

- Number of positive examples on $R_B(A)$: $n^+(R_B(A)) = |E^+(R_B(A))|$
- Number of negative examples on $R_B(A)$: $n^-(R_B(A)) = |E^-(R_B(A))|$
- Number of covered examples on $R_B(A)$: $n(R_B(A)) = |E(R_B(A))|$
- The weight of a rule $R_B(A)$: $\omega(R_B(A)) = \frac{n^+(R_B(A))}{n(R_B(A))}$.

3.1 Infering with fuzzy rules

The inference process consists of determining wich is the classification of an example, given a set of rules. When we work with fuzzy rules, we can distinguish between two different situations: crisp and fuzzy consequent. In the first case, the class of the rule with the best adaptation between example and antecedent of the rule is the output,.In the second case the output combines information about different rules and we can use several approaches [10]

SLAVE can learn and infer in both situations using as compatibility measure the U function. However, the crisp consequent inference needs a more detailed description since SLAVE tries to solve the conflict problems. Thus, given an example e and a set of fuzzy rules $R(A^i)$, the rule that determines the classification of this example is $R(A^c)$, where c satisfies that:

$$U(e, A^c) \geq U(e, A^i) \; / i = 1 \ldots p$$

where p is the number of rule of the rule set.

When we work with crisp consequent variables, can appear two or more antecedent of rules from different classes obtaining the same best degree of compatibility with the example (conflict problem). In the literature has been proposed different solutions for solving this problem [3, 24].

In order to solve the conflicts that should appear in the inference process, SLAVE uses the weight of the rule previously defined. Thus, when a conflict appears, first, the inference module of SLAVE selects the rule with maximum weight value because a rule with a value of ω close to one, indicates a good accuracy of this rule on the training set.

When two or more rules with the same weight are actived, then the inference process selects between them, the rule that covered the least number of examples in the learning process. If there are two or more rules in this situation, is actived the rule that first was learnt in the learning process.

In summary, the conflicts are solved in SLAVE through use of the following ordered criterium:

i) to break a tie using the weight of the rule.

ii) to break a tie selecting the rule that covered the least number of examples in the learning process.

iii) to break a tie using the first rule learned.

4 The SLAVE learning algorithm

SLAVE (Structural Learning Algorithm in Vague Environment) is an inductive learning algorithm that uses genetic algorithms and fuzzy logic concepts. This system is able to learn the structure of the rule and cope with fuzzy information. When we refer to obtaining the structure of the rule, we want to find the predictive variables that are needed for representing a concret value of the consequent variable. In this sense, SLAVE can be classified among the algorithms that try to learn the partial relevance of the initial attributes, according to the classification proposed by Michalski in [27].

The algorithm begins with a simple description of the problem: the consequent variable (concept variable) and the set of all the available antecedent variables for generating the rules that describe the consequent variable. The learning algorithm, using this description and a set of examples, will decide for each class and each rule which are the variables needed to describe the concept (feature selection), and the rest will be eliminated. Therefore, the goal of SLAVE is extracting a set of fuzzy rules that represents the relationship between the consequent variable and the predictives variables, obtaining for each rule only the predictive variables that are necessary for representing the value of the consequent variable.

The SLAVE learning algorithm, given a training example set E and a value B of the consequent variable, tries to find the best combination of values A=$(A_1, \ldots, A_p)$ of the antecedent variables for describing the examples in the E set. This task is done in the following way: the best rule that represents the examples of the B class is selected. This rule is appent to the rule set. If more rules are needed for representing the examples of the B class, the examples cover for the previous obtained rules are eliminated from the training set and the process is repeated until no more rule are needed. When all rules for a class have been leant, the same process is repeated for the rest of the class. A graphical description of this process is shown in Figure 2.

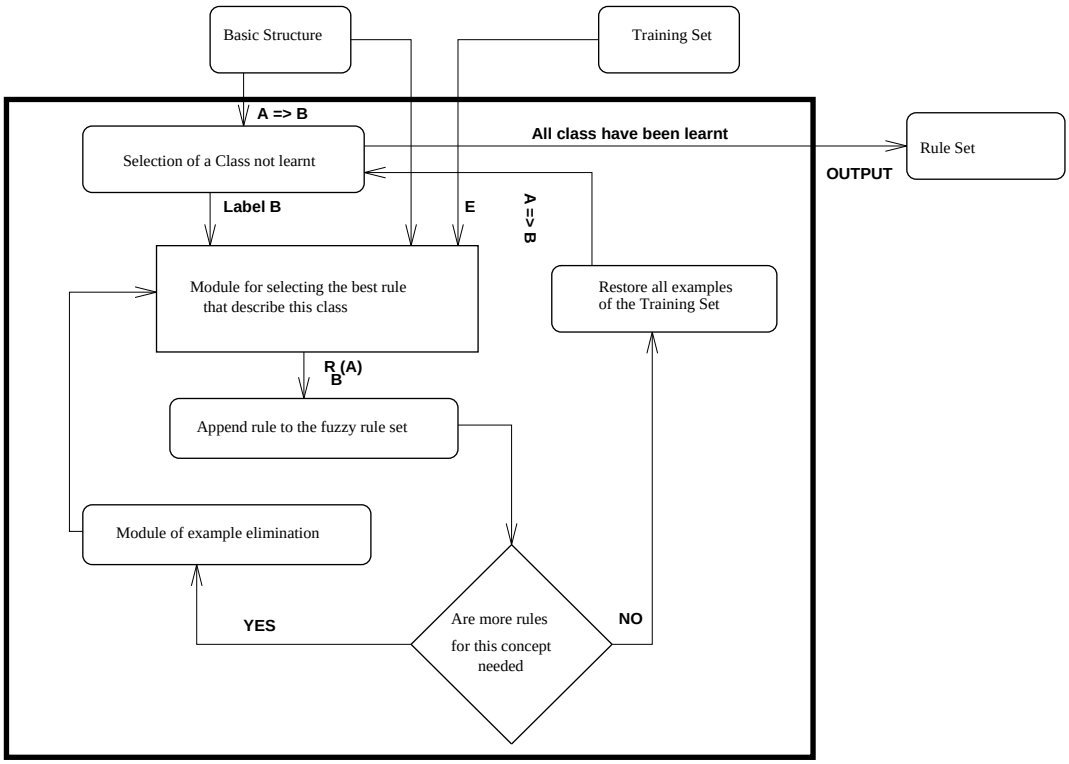


Figure 2: SLAVE learning process

In the next three subsection, we describe briefly the most important components of this process. The first subsection describes the theoric model in which is based SLAVE. The second subsection describes the genetic

algorithm used by SLAVE. The third subsection is devoted to show the module of example elimination.

4.1 The best rule ≡ consistent and complete

In the SLAVE learning process, to find the best rule consists of determining the best combination of values of the antecedent variables, given a fix value of the consequent variable and a set of examples. The best rule concept uses a simple quantitative criterion, the best rule will be the rule covering the maximum number of examples. But this criterion has problems if we do not restrict the set of possible rules.

The classical learning theory proposes a condition that must be verified for the set of rules that are obtained by a learning algorithm. These conditions that provide the logical foundation of the algorithms for concept learning from examples are called *consistency condition* and *completeness condition* [27].

Definition 4.1 *The completeness condition states that every example in some class must verify some rule from this class.*

Definition 4.2 *The consistency condition states that if an example satisfies a description of some class, then it cannot be a member of a training set of any other class.*

These definitions are associated on the whole set of rules. SLAVE obtains the set of rules that describes the system, extracting one rule in each iteration of the learning process. For this reason, we need to define these concepts on each rule. Moreover, we are not interested in proposing hard definitions on fuzzy problems, thus, we popose a degree of completeness and a degree of consistency, both definitions use the concepts proposed in section 3.

Definition 4.3 *The degree of completeness of a rule $R_B(A)$ is defined as*

$$\Lambda(R_B(A)) = \frac{n^+(R_B(A))}{n_B}$$

where $n_B = \sum_{i=1}^{m} U(e_i, B)$ and m is the number of example in the training set.

With respect to the soft consistency degree [16] is based on the possibility of admiting some noise in the rules. Thus, in order to define the soft consistency degree we use the following set:

$$\Delta^k = \{R_B(A)/n^-(R_B(A)) < k\ n^+(R_B(A))$$

that represents the set of rules having a number of negative examples strictly less than a porcentage (depending on k) of the positive examples.

Definition 4.4 *The degree in which a rule satisfies the soft consistency condition is*

$$\Gamma_{k_1 k_2}(R) = \begin{cases} 1 & \text{if } R \in \Delta^{k_1} \\ \frac{k_2 n_E^+(R) - n_E^-(R)}{n_E^+(R)(k_2 - k_1)} & \text{if } R \notin \Delta^{k_1} \text{ y } R \in \Delta^{k_2} \\ 0 & \text{otherwise} \end{cases}$$

where $k_1, k_2 \in [0,1]$ and $k_1 < k_2$, and $n_E^-(R)$, $n_E^+(R)$ are the number of positive and negative examples to the rule R.

This definition uses two parameters, k_1 is a lower bound of the noisy threshold and k_2 is a upper bound of the noisy threshold.

Thus, SLAVE selects the rule that simultaneously verifies in a high degree the completeness and soft consistency conditions. Therefore, the rule selection in SLAVE can be solved by the following optimization problem.

$$(1) \qquad \max_{A \in D} \Lambda(R_B(A))\ \Gamma_{k_1 k_2}(R_B(A))\}$$

where $D = P(D_1) \times P(D_2) \times \ldots \times P(D_n)$.

In SLAVE we use a genetic algorithm for solving this search problem. This genetic algorithm is described in the next section.

4.2 The Genetic Algorithm of SLAVE

The genetic algorithm of SLAVE was initially proposed in [14] and later described in [16, 18]. Now, we show briefly the main elements of this algorithm in order to better the understand the modified version proposed in the next section.

The genetic algorithm of SLAVE uses an elitist model and for its description it is necessary to define its five components: genetic representation of the rules, generation of the initial population, the fitness function, the genetic operators and the termination condition.

- **Representation of an element of the population**

  For the genetic representation we use the following binary representation described in [14]: if the database has n antecedent variables $X_1, \ldots, X_n$ and each one of them has a fuzzy domain D_i associated with m_i component, then we use the following method of coding of any elements of $P(D_1) \times \ldots \times P(D_n)$ a vector of $m_1 + \ldots + m_n$ zero-one components, such that,

$$component(m_1 + \ldots + m_{r-1} + s) = \begin{cases} 1 & \text{if the s-th element in the domain } D_r \\ & \text{is a value of the } X_r \text{ variable} \\ 0 & \text{otherwise} \end{cases}$$

  Example 2 *Let us suppose we have three variables X_1, X_2 y X_3, such that the fuzzy domain associated with each one is shown in Figure 3. In this case, the vector 1010010111 represents the following antecedent*

$$X_1 \text{ is } \{A_{11}, A_{13}\} \text{ and } X_2 \text{ is } \{A_{23}, A_{25}\} \text{ and } X_3 \text{ is } \{A_{31}, A_{32}\}.$$

  Since X_3 takes every one of the elements of the domain D_3, the antecedent above is equivalent to

$$X_1 \text{ is } \{A_{11}, A_{13}\} \text{ and } X_2 \text{ is } \{A_{23}, A_{25}\}.$$

- **Generating the first population**

  The generation of the initial population consists of selecting examples at random from the class that must be learned and to obtain for each one of them, the most specific antecedent that best covers it. This antecedent is made-up by only one label for each antecedent variable. The label for each antecedent variable is the one that gives the highest degree of membership for each component in the example.

  Example 3 *Let X_1, X_2 y X_3 be variables with an associated domain corresponding to Figure 3, and let (r1,r2,r3) be the randomly selected example from the training set for a class. The most specific antecedent that best represents it is:*

$$X_1 \text{ is } A_{13} \text{ and } X_2 \text{ is } A_{23} \text{ and } X_3 \text{ is } A_{21}$$

  the binary representation of which is 0010010010.

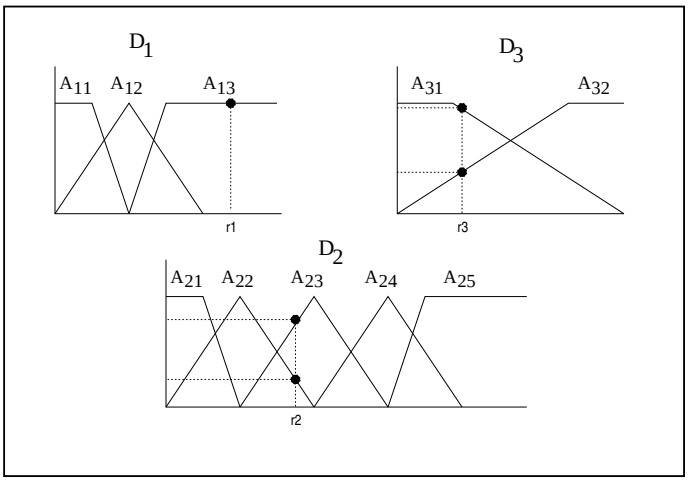


Figure 3: Domains of the X_1, X_2, X_3 variables

When there are several options, we select one of them at random.

This selection of the initial population is similar to the one used by the AQ algorithms [27], because for generating the initial antecedent for a class, we take examples of this class in the training set as a starting point, and the genetic process could be considered as a generalization process over these antecedents selected.

- **Evaluation Function (FITNESS)**

  The goal of the genetic algorithm is finding the best rule in the space of the rules that verifies to have a high degree of completeness and a high degree of consistency given a set of examples. Under the criterion from Section 2.1 and given a rule $R_B(A)$ and a set of training examples E, we define the evaluation function as

$$fitness(R_B(A)) = \Lambda(R_B(A)) * \Gamma_{k_1k_2}(R_B(A))$$

  where $\Gamma_{k_1k_2}(R_B(A))$ is the degree with which the rule $R_B(A)$ satisfies the k_1k_2-consistency condition

  and $\Lambda(R_B(A))$ is its degree of the completeness.

  The best rule is the one that maximize this fitness function, that is, the genetic process returns the rule covering the highest number of positive examples verifying the soft consistency conditions.

- **Genetic Operators**

  - **Ranking selection operator**. This is a selection model that sorts the elements of the population using its fitness valuation, and a probability selection is assigned for each position of the population.
  - **Crossover operator over two points**. This type of crossover establishes two cutoff point between two elements in the population and exchange the central segment like the one is shown in Figure 4.
  - **Mutation operator.** In the SLAVE genetic algorithm the classical mutation operator is implemented. This operator changes one gen of an element in the population with a certain probability.

- **Termination Condition**

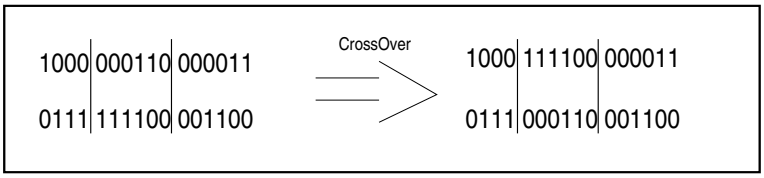


Figure 4: Crossover operator over two points

In the implementation of the termination condition we distinguish between the class that has at least one rule and the class that does not have any rules. We make a more exhaustive search when we wish to find the first rule of a class and relax this process of search when we have some rules for a class.

The genetic algorithm ends returning the best rule of the last population if one of the next conditions is verified.

- the number of iterations is greater than a fixed limit.
- the fitness function of the best rule in the population does not increase the value during at least a fixed number of iterations and rules with this consequent have already been obtained.
- there are no rules with this value of the consequent obtained before, but the fitness function does not increase the value during a fixed number of iterations and the current best rule can eliminate at least one example from the training set.

4.3 Module of Example Elimination: The Concept of λ-covering

We need to establish a condition for determining when a set of rules for a class is sufficient for representing to a system. When we work with crisp examples and rules, the implementation of this condition is very easy, but when we work with fuzzy examples and rules problems appear because the adaptation between a rule and an example is not crisp, that is, an example is covered by a rule to a certain degree. In [16] the following definition of covering between an example and a rule is proposed.

Definition 4.5 *Let R be a rule and E a set of examples. The subset of E representing the examples λ-covered by R is the λ-cut of $E^+(R)$. λ will be called the covering parameter.*

We said previously that the algorithm must repeat the process of searching for the best rule and eliminating the examples covered by it while new rules are needed for determining a value of the consequent variable. Really, this condition is equivalent to the completeness condition on the fuzzy rule set and it is very difficult to satisfy when we work with noisy or fuzzy examples. Thus, we propose a condition for considering whether a class has been sufficiently learnt. This condition takes into account two considerations:

a) all the examples of the class are adequately covered by the obtained rules.

b) the rules do not cover adequately the examples of the class, but the learning algorithm is not able to obtain a new useful rule.

In the following definition we describe formally this condition:

Definition 4.6 *Given a set of rule RULES=$\{R_B(A^1), R_B(A^2), \ldots, R_B(A^j)\}$ previously obtained for the B class by the algorithm, and a new rule $R_B(A^{j+1})$. We say that the set RULES is sufficient for representing the B class if is verified one of the following conditions:*

a) $\frac{\sum_{t=1}^{m} \max_{i=1}^{j}\{U(e_t,A^i) * U(e_t,B)\}}{n_B} = 1$

b) i) fitness($R_B(A^{j+1})$)=0

ii) the cardinality of the λ-cut of $E^+(R_B(A^{j+1}))$ is zero, that is,

$$|E^+(R_B(A^{j+1}))_\lambda| = 0 \; \forall e \; \in \; E_{j+1}$$

where m is the number of examples from the training set and n_B is the number of examples from the B class in the training set, λ is the covering parameter and E_{j+1} is the training set without the examples covered in a degree greater than λ by the RULES set.

The first condition determines if RULES covers all the examples of its class. This condition is equivalent to the classical completeness conditions applied on only one class.

The second condition checks when the learning algorithm is not able to obtain a new useful rule. The criterion of the learning is always to obtain the best rule, and if it is not useful, we decide that the previous rule set for this class is enough. This second condition has two component: the first one establishs that the next best rule ($R_B(A^{j+1})$) must not be included in the RULES set because it has zero degree of completeness or/and zero degree of soft consistency. The second one determines that the next rule must not be included in the RULES set because it is not able to cover any example in a degree greater or iqual than λ.

In the learning component previously described, the example elimination module eliminates from the training set those examples that are λ-covered by the last rule obtained in the module for selecting the best rule and checks whether the rule set obtained is sufficient for describing the current class using the weak completeness condition. In the last two conditions, the last rule obtained is not included in the rule set.

5 A new proposal of SLAVE learning algorithm

In the previous sections, we have described briefly the SLAVE learning algorithm. In works, such as [15, 18], we have tested this system with other learning algorithms on several databases and we shown that its results are comparable with them. But, experimentally, we are detected some problems, such as: the diversity in the population of the GA, rules that take into account too much predictive variables, the elimination of covered examples as rule penalization model in the genetic approach is not quite adecuated in problems that we want to describe using linguistic variables, etc.

The three next subsections are devoted to improve the behaviour of SLAVE, solving the previous problems.

5.1 Appending new genetic operators to SLAVE

In this section, we study experimentally some modifications from the genetic algorithm for defining a more appropiate configuration of the module of the rule selection. This modification consists of introducing new genetic operators with the following two intentions:

a) Improving the diversity in the genetic population.

b) Increasing the understanding of the rules that are obtained.

Specifically, we study the inclusion of the following new genetic operators in the original system:

- **Rotation operator**. This is a modified version of the traditional inversion operator that we propose in order to include a higher diversity in the searching problem. This operator takes a cutoff point in an element of the population and changes the position of the two segments as is shown in Figure 5.

- **OR operator** This is a special crossover operator that tries to represent a generalization process between elements of the population. The operation is very similar to the crossover operator over two points, e.i., two cutoff point are defined between two elements in the population and the central segment is combined with a bit to bit OR operation as is shown in Figure 6.

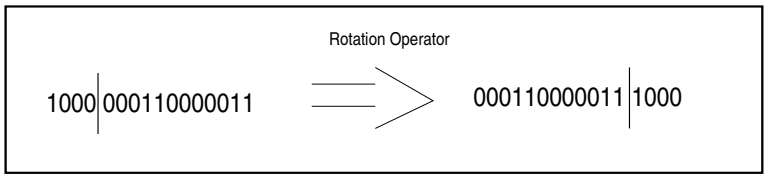


Figure 5: Rotation operator

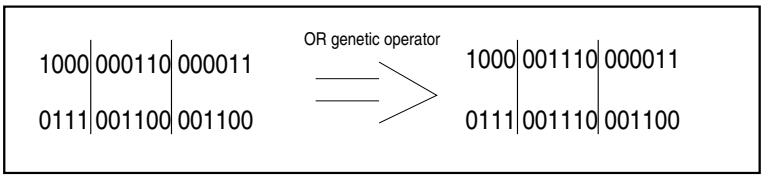


Figure 6: OR operator

- **Generalization operator**. We propose a specific operator that tries to make the rules returned by the learning process clearer and more understandable. This operator generalizes the rules of the population using a special criterion and we show its behaviour with the next example.

  Example 4 *Let X_2 be the variable for which the domain is defined the Figure 3. Let us suppose that the genetic algorithm must to select between two posible antecedents, for example,* **00111** *and* **00101**, *such that* **fitness(00111)=fitness(00101)**.

  The first value may be expressed as

  $$\ldots \textit{ and } X_2 \textit{ is } \{A_{23}, A_{24}, A_{25}\} \textit{ and } \ldots$$

  but, we can rewritten this antecedent as

  $$\ldots \textit{ and } X_2 \textit{ is higher than or equal to } A_{23} \textit{ and } \ldots$$

  while the second value only may be expressed as

  $$\ldots \textit{ and } X_2 \textit{ is } \{A_{23}, A_{25}\} \textit{ and } \ldots$$

  We need to decide between both values. We can see that the first value presents a more understandable description from a descriptive point of view but the GA selects randomly one of them.

  In the genetic algorithm, we will prefer the solutions that correspond to decisions more generical. Therefore, we use a genetic operator for adding this behaviour in the genetic algorithm.

  Before describing the criterion, we need to introduce some previous concepts.

  Definition 5.1 *We say that a variable of an antecedent is stable if the binary genetic representation of its value has only one continuous sequence of ones.*

  Definition 5.2 *We say that a variable of an antecedent is unstable if the binary genetic representation of its value has more than one continuous sequence of ones.*

  Definition 5.3 *Given a binary genetic representation of a value, we shall call each continuous sequence of zeros of its representation an unstable zone.*

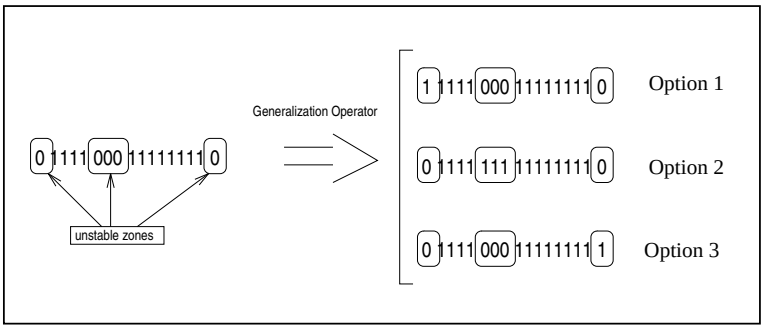


Figure 7: Generalization Operator

The generalization operator tries to obtain stable variables by eliminating its unstable zones. This operator could be described in the following way:

1. A variable of an antecedent of the population is selected at random.
2. If this variable is unstable do the following
   2.1 Detect its unstable zones and randomly select one of them.
   2.2 Change the unstable zone selected for a continuous sequence of ones.
   2.3 Calculate the value of the new element using the fitness function. We include the new element if its fitness is equal to or greater than the original element.

Figure 7 shows the unstable zones of a value. Furthermore, in this Figure three options of the generalization operators are shown.

For testing the behaviour of SLAVE with this new set of operators, we have made different experimental tests on the following databases and all possible combinations between the new operators. In Table 2, we present the most significant combinations. We establish the goodness of each combination over three parameters: accuracy, number of rules and time needed for learning. The databases used in the experimental work are the following:

- The IRIS database [11]. In this database we try to classify three different classes of plants using four predictive variables, using 150 examples. All the predictive variables have a continuous domain and we have considered five uniformly distributed linguistic labels for discretizing each range.
- The INFARCTION database [22, 15]. This database is made up by 14 different variables for determining the nominal diagnostic variable. There are 2 nominal variables and the rest of them are continuous. These variables are divided into two different type of tests, morphological methods and biochemical analysis. These tests are accepted as the best markers for the postmortem diagnosis of myocardial infarction. The databases has 78 examples and contains missing values. We have considered seven uniformly distributed linguistic labels for discretizing the continous variables.
- WINE recognition data. These data are the results of a chemical analysis of wines from the same region but different types of grapes, using 13 continuous variables and 178 examples. We have used seven uniformly distributed linguistic labels for discretizing the range of each variable.
- The IONOSPHERE data [30]. This radar data was collected by a system in Goose Bay, Labrador. This system consists of a phased array of 16 high-frequency antennas with a total transmitted power on the order of 6.4 kilowatts. The targets were free electrons in the ionosphere. The database is composed by 34 continuous attributes plus the class variable, using 351 examples. In the same way that the previous databases, we have used seven uniformly distributed linguistic labels for discretizing the range of each variable.

For each database, we have produced five partitions in training and test sets (70% and 30% respectively). For each database, we have executed SLAVE using the parameters described in Table 1.

Parameter	Value
maximum number of iterations	1000
population size	20
mutation probability	0.1
crossover probability	0.6
k_1	0
k_2	1

Table 1: SLAVE's parameters

		IRIS	INFARCTION	WINE	IONOSPHERE
the original SLAVE	accuracy	95.72	83.22	88.54	82.41
	rules	4.2	11.8	21.8	48.4
	time	1.00	1.00	1.00	1.00
only OR	accuracy	95.72	83.41	89.13	83.12
	rules	4.2	8.6	18.3	42.4
	time	0.94	0.36	0.48	0.60
OR and Generalization	accuracy	95.72	86.4	90.44	84.31
	rules	4.4	6.8	16.6	40.8
	time	0.46	0.16	0.29	0.41
OR and Rotation	accuracy	95.72	89.04	93.32	84.34
	rules	4.2	8.4	16	46
	time	0.46	0.17	0.28	0.44
OR, Generalization, Rotation	accuracy	95.72	90.03	93.83	84.70
	rules	4.4	7.4	16.2	37.2
	time	0.45	0.16	0.27	0.39

Table 2: Experimental results

Table 2 shows the average of the five partitions on each databases and each combination of genetic operators. For comparing the different options, we have used three parameters: the accuracy on the test set (accuracy), the number of rules contains in rule set (rules) and the rate between the time that each combination needs for learning and the time that the original SLAVE needs for learning the problem (time). For example, A value of 0.60 in this last parameter, indicates that the average for that combination has needed a 60% of the time that the original SLAVE needed for obtaining the rule set. With these results we can see that the combination of the three new operators obtains the best results in accuracy and time over all databases. With respect to the number of the rules the combination of the three new operators is the best, except for the IRIS databases, but with a little difference.

We can conclude that the combination of the three new operators improves the performance of the learning algorithm. Therefore, from now, we include this new operators as components of the GA from SLAVE and the whole genetic operator set is

- Mutation.
- Crossover over two points.

- OR.
- Rotation.
- Generalization.

5.2 A multiobjetive fitness function

The goal of the genetic algorithm is finding in each step the most consistent and complete rule given the available set of examples. But, if we want to learn the structure of a rule then we must take into account in the evaluation of a rule its simplicity and its understandable. In order to reward the effect of the generalization genetic operator, we propose a new multiobjetive fitness function [12] that includes the previous one and two new components:

a) **Information about the simplicity of the rule.** This component is calculated through the number of irrelevant variables in the rule using the following expression:

$$S(R_B(A)) = \frac{1 + Irrelevant(A)}{1 + p}$$

where Irrelevant(A) is the number of the irrelevant variables of the antecedent of the rule and p is the number of the initial destriptors.

This objetive is too relation with to learn the structure of the rule, therefore, with this component we want descriptions of the classes that imply the miminum number of initial destriptors.

b) **Information about the most understandable rules.** This component is associated to the stable concept defined before and it has the following expression:

$$CP(R_B(A)) = \frac{1 + Stable(A)}{1 + p}$$

where Stable(A) is the number of the stable variables of the antecedent of the rule and p is the number of the initial destriptors.

Thus, now each element in the population of the genetic is evaluated under three differents point of view. The evaluation of each one of them is a value in [0,1]. Therefore, we can consider that the fitness function is a mapping from the space of possible antecedent ($D = P(D_1) \times P(D_2) \times \ldots \times P(D_p)$.) to $[0,1]^3$,

$$fitness : D \longrightarrow [0,1]^3$$

$$fitness(R_B(A)) = (\Lambda(R_B(A)\ \Sigma_{k_1k_2}(R_B(A)), S(R_B(A)), CP(R_B(A)))$$

where the first component is the initial fitness function, the second component is the S function and the last one is the CP function. For searching the best rule we use the lexicographical order in $[0,1]^3$, that is, the best rules is that one that has the maximum value of $\Lambda(R_B(A))\ \Sigma_{k_1k_2}(R_B(A))$. If two or more rules have the maximum value, then the best rule is that one, between them, that has the maximum value of $S(R_B(A))$. If there are more of one rule that have the maximum values in these two objetives, then the best rule is that one that has the maximum value of $CP(R_B(A))$. When these three objetives are not sufficient for breaking to tie between the rules, then one of them is randomly selected.

5.3 A new way for learning with the iterative approach

As was shown previously, the iterative model consists of using a genetic algorithm for obtaining a single rule that represents a partial solution to the problem that we want to learn. For obtaining the global solution, is needed to include the GA into an iterative schema. The obtained rules are penalized for do not select them again. In some learning systems, such as SIA [33] or the original SLAVE that use this iterative approach, the penalization of the rule is done removing the examples covered by the rule set from the training set.

This penalization criterion can cause some problems, when we work with crisp consequents, in the following sense: Let suppose that T rules of a class have been learnt and we wish to learn the T+1 rule for the same class. The information that the system has about the T rules, are the examples from the training set that were not covered by them, but for the system is not known as the T rules affect on the examples from other classes. Furthermore, when we work with fuzzy rules and examples, this problem increases since the examples are covered by a rule in a degree. This can cause unexpected interactions between the rules during the inference process.

For solving this anomalous behaviour, we propose to keep the information relative to the previously learnt rule and use a new definition of the concepts of the positive and negative example. Thus, we establish theconcepts of maximum positive and negative covering degree of an example with respect to the learnt rules.

Definition 5.4 *Let $E^B = \{e_1^B, e_2^B, \ldots, e_s^B\}$ be a set of s training examples that describes to B class, Let $RULES = R_B \bigcup R_{\overline{B}}$ be the set of the obtained rules at the moment, where R_B is the set of the learnt rules from the B class and $R_{\overline{B}}$ is the set of the learnt rules from the rest of the classes.*

- *We define* **the maximum positive covering degree of an example over** R_B *as:*

$$Q_B(R_B, e_i^B) = \max_{R_B(A^j) \in R_B} \{U(e_i^B, A^j)\}$$

- *We define* **the maximum negative covering degree of an example over** $R_{\overline{B}}$ *as:*

$$Q_{\overline{B}}(R_{\overline{B}}, e_i^B) = \max_{R_{\overline{B}}(A^j) \in R_{\overline{B}}} \{U(e_i^B, A^j)\}$$

The previous definitions establish for each example from the training set the maximum degree in which the rules of its class cover this example (Q_B) and the maximum degree in which the rest of the rules cover it ($Q_{\overline{B}}$9 respectively. In the iterative approach the values of Q_B and $Q_{\overline{B}}$ are modified each times that a new rule is included in the rule set.

From these definitions, we can to redefine the concept of positive example from the a rule in the following sense: we only consider an example as positive for a new rule if the compatibility between the example and this rule is better than the maximum positive and the maximum negative covering degree of this example. In a similar way, we only consider an example as negative, if the compatibility between the example from other class and the new rule is greater or iqual than the maximum positive covering degree for this example.

Definition 5.5 *Let $E = E^B \bigcup E^{\overline{B}}$ be the training set, where E^B is the set with all the examples that describes the B class and $E^{\overline{B}}$ is the set with all the examples that describes the rest of the classes. Let $RULES = R_B \bigcup R_{\overline{B}}$ be the whole set of the obtained rules, where R_B is the set with all rules of the B class and $R_{\overline{B}}$ is the rule set of the rest of the classes. Let $R_B(A)$ be a new rule for the B class do not included in R_B set.*

- *We define* **the fuzzy set of positive example of the rule** $R_B(A)$ *as:*

$$E_Q^+(R_B(A)) = \{(e_i, Gr^+(R_B(A), e_i))/e_i \in E^B\}$$

where

$$Gr^{+}(R_B(A), e_i) = \begin{cases} U(e_i, A) & \text{if } U(e_i.A) > Q_B(R_B, e_i) \text{ and } U(e_i, A) > Q_{\overline{B}}(R_{\overline{B}}, e_i) \\ 0 & \text{otherwise} \end{cases}$$

- *We define* **the fuzzy set of negative example of the rule** $R_B(A)$ *as:*

$$E_Q^{-}(R_B(A)) = \{(e_i, Gr^{-}(R_B(A), e_i))/e_i \in E^{\overline{B}}\}$$

where

$$Gr^{-}(R_B(A), e_i) = \begin{cases} U(e_i, A) & \text{if } U(e_i, A) \geq Q_B(R_B, e_i) \\ 0 & \text{otherwise} \end{cases}$$

With these definitions, we can define the number of positive and negative examples using the cardinal of each one of these fuzzy sets in a similar way that they are proposed in the section 3. The following example clarifies these definitions.

Example 5 *Let* $E = \{e_1^{B_1}, e_2^{B_1}, e_3^{B_1}, e_1^{B_2}, e_2^{B_2}, e_3^{B_2}, e_4^{B_2}\}$ *be the training set for describing two different classes. Let RULES be the set of all rules already obtained. For this set of rule, the values of the maximum positive and negative covering degree (*Q_B *and* $Q_{\overline{B}}$ *respectively) are shown in Table 3.*

	$e_1^{B_1}$	$e_2^{B_1}$	$e_3^{B_1}$	$e_1^{B_2}$	$e_2^{B_2}$	$e_3^{B_2}$	$e_4^{B_2}$
Q_B	0.3	0.8	0.3	0.1	0.7	0.0	0.0
$Q_{\overline{B}}$	0.2	0.5	0.8	0.4	0.3	0.5	0.6

Table 3: The covering degree for each example in E over the RULES set.

Under this conditions, we want to evaluate a new rule $R_{B_2}(A)$ *from the* B_2 *class, where* $R_{B_2}(A) \notin RULES$. *The compatibilities between this rule with the examples in the training set are shown in Table 4.*

	$e_1^{B_1}$	$e_2^{B_1}$	$e_3^{B_1}$	$e_1^{B_2}$	$e_2^{B_2}$	$e_3^{B_2}$	$e_4^{B_2}$
$U(e_i, A)$	0.4	0.6	0.4	0.6	0.5	0.7	0.3

Table 4: The compatibility between each example and $R_{B_2}(A)$.

For obtaining the sets $E_Q^{+}(R_{B_2}(A))$ *and* $E_Q^{-}(R_{B_2}(A))$, *we compare the results of Table 3 and Table 4 for each example.*

- *The examples* $e_1^{B_1}$ *and* $e_3^{B_1}$ *are considered as negative examples for the rule, because both are from the* B_1 *class and their maximum positive covering degrees are less than their compatibilies between* $R_{B_2}(A)$ *and these examples.*
- *The example* $e_2^{B_1}$ *has compatibility with the rule and it is from the* B_1 *class, but it is not considered as negative example because its maximum positive covering degree is greater than its compatibility with the rule.*
- *The examples considered as positives are* $e_1^{B_2}$ *and* $e_3^{B_2}$, *because their compatibilies with the rule, improve their positive covering degrees and they are greater than their maximum negative covering degrees.*
- *The examples* $e_2^{B_2}$ *and* $e_4^{B_2}$ *are not considered as positive examples. The first of them, because its compatibility with the rule is not greater than its maximum positive covering degree. The second of them, because its maximum negative covering degree is greater than its compatibility with the rule.*

Therefore, the number of positive and negative examples to the rule $R_{B_2}(A)$ are :

$$|E_Q^+(R_{B_2}(A))| = |\{(e_1^{B_2}, 0.6), (e_3^{B_2}, 0.7)\}| = 1.3$$

$$|E_Q^-(R_{B_2}(A))| = |\{(e_1^{B_1}, 0.4), (e_3^{B_1}, 0.4)\}| = 0.8$$

Now, let suppose that the rule $R_{B_2}(A)$ is the best rule for this execution and it must be included in the RULES set. Then, the maximum positive and negative covering degree are modified as shown Table 5.

	$e_1^{B_1}$	$e_2^{B_1}$	$e_3^{B_1}$	$e_1^{B_2}$	$e_2^{B_2}$	$e_3^{B_2}$	$e_4^{B_2}$
Q_B	*0.7*	*0.8*	*0.3*	*0.6*	*0.7*	*0.7*	*0.3*
$Q_{\overline{B}}$	*0.8*	*0.6*	*0.8*	*0.4*	*0.3*	*0.5*	*0.6*

Table 5: Modification of the covering degree.

Seeing Table 5 we can determine the examples that will be correctly classified using the RULES set. If the maximum positive covering degree for an example is greater than the maximum negative covering degree, then this example will be correctly classified by the rule set. From opposite of this, if the maximum negative covering degree is greater than the maximum positive covering degree then this example will not be correctly classified. When both maximum covering degrees for an example have got the same value, then we have a conflict problem and its correct classification depend on the mecanism for solving conflicts that we have proposed in section 3.1.

This modification in the measurement of the positive an negative examples causes two improvements:

a) The examples are only considered as positive or negative to a rule, when they cause a significative change over theirs classification, taking into account the learnt rules.

b) The genetic algorithm has a greater tendency to generalize the rules than the original genetic algorithm of SLAVE. This tendency associated to the new genetic operators defined previously causes that the number of rule necessary is reduced for describing the system and the understanding of each one of them is improved.

6 Testing the new version of SLAVE

Now, we will compare the new version of SLAVE with its old version, and with two well-knwon learning algorithm C4.5 and C4.5rules that was developed in [29]. We use the same databases, that was proposed in Section 5.1 and with the same parameters that we can seen in Table 1. For comparing the two versions of SLAVE we have used three parameters: accuracy on the test set (accuracy), the number of the rule obtained (rules) and the rate between the time that each version of SLAVE needs for learning the problem and the time that the original SLAVE needs for the same problem (time) We note **SLAVE[1]** the original SLAVE learning algorithm including only the new three genetic operators and we note **SLAVE[2]** the original SLAVE algorithm including the new three genetic operators, the new multiobjective fitness function and the new method for penalizing examples. In Table 6 are shown the results obtained.

From the results, we can deduce the following conclusions:

a) SLAVE[1] and SLAVE[2] cause a great improvement on the three studied parameters with respect to the original SLAVE learning algorithm.

b) The results between SLAVE[1] and SLAVE[2] are very similar in accuracy, but there are a great different between both when we compared the time and the number of rules needed for obtaining the solution. SLAVE[2] presents a better behaviour with respect to these two parameters.

		IRIS	INFARCTION	WINE	IONOSPHERE
C4.5	accuracy	92.7	81.4	93.2	86.5
C4.5rules	accuracy	94.4	82.2	93.5	86.3
	accuracy	95.7	83.2	88.5	82.4
the original SLAVE	rules	4.2	11.8	21.8	48.4
	time	1.00	1.00	1.00	1.00
	accuracy	95.7	90.0	93.8	84.7
SLAVE[1]	rules	4.4	7.4	16.2	37.2
	time	0.45	0.16	0.27	0.39
	accuracy	95.7	89.3	93.8	85.7
SLAVE[2]	rules	4.4	6.8	7.6	6.2
	time	0.32	0.11	0.16	0.31

Table 6: Table of results

c) The accuracy of SLAVE[2] is better than C4.5 and C4.5rules in three of the four databases. In the fourth database, the results among the learning algorithms are very close, but are better C4.5 and C4.5rules. We can conclude that the new modifications included in the SLAVE learning algorithm are good in comparation with these learning algorithms.

7 Conclusions

SLAVE is a learning algorithm that uses the iterative approach for learning with GA. The main element of SLAVE is its rule model that permit it to determine the structure of the rule. Another important feasure of this learning algorithm is that can work with data vaguely defined and with missing values.

In this work, we have presented some modifications for improving the SLAVE behaviour. These modifications mainly affect to the component of the GA. Thus, we have introduced three genetic operators, two of them for obtaining more understandable descriptions and the other for increasing the diversity level in the population. These operators have shown: improve the predictive accuracy, reduce the number of rules and reduce the time needed for obtaining the rule set.

Furthermore, we have introduced a multiobjetive fitness function for obtaining more understandable descriptions and we have proposed a new model for penalizing rules that it is more adequated than the model of elimination of covered examples for working with the iterative approach when we use predictive linguistic variables.

SLAVE with these modifications has a better behaviour than the original SLAVE with respect to the three studied parameters (accuracy, rules and time) and its results in accuracy are comparable with other classical learning algorithms.

References

[1] Barone J.M., *Pruning Fuzzy Decision Trees*. Proceedings of the 2nd International Conference on Fuzzy Logic and Neural Networks, pp. 321-324 (1992).

[2] Carse, B., Fogarty, T.C., *A Fuzzy Classifier System Using the Pittsburgh Approach*. Parallel Problem Solving from Nature - PPSN III, Springer-Verlag, pp. 260-269. (1994).

[3] Cordón O., del Jesús M.J., Herrera F., López E., *Selecting Fuzzy Rule-Based Classification Systems with Specific Reasoning Methods Using Genetic Algorithms*. To appear in Proc. IFSA'97. (1997).

[4] Cordón O., Herrera F., *A Three-Stage Process for Learning Descriptive and Approximative Fuzzy Logic Controller Knowledge Bases from Examples*. To appear in Intl. Journal of Approximate Reasoning. (1997).

[5] Clark P., Niblett T., *Learning If Then Rules in Noisy Domains.* TIRM 86-019, The Turing Institute, Glasgow (1986).

[6] De Jong K.*Learning with Genetic Algorithms: An overview.* Machine Learning 3, pp. 121-138 (1988).

[7] De Jong K.A., Spears W.M., Gordon D.F., *Using Genetic Algorithms for Concept Learning.* Machine Learning, 13, pp. 161-188. (1993).

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

[9] Dorigo M., *New Perspectives about Default Hier Formation in Learning Classifier Systems.* Technical Report No. 91-002. Dipartamento di Elettronica, Politecnico di Milano Italy (1991).

[10] Driankov D., Hellendoorn H., Reinfrank M., *An Introduction to Fuzzy Control.* Springer-Verlag (1993).

[11] Fisher, R.A. *The Use of Multiple Measurements in Taxonomic problems.* Annual Eugenics, 7, Part II, 179-188. (1936).

[12] Fonseca C.M., Fleming P.J., *An Overview of Evolutionary Algorithms in Multiobjetive Optimization.* Evolutionary Computation, vol. 3, pp. 1-16 (1995).

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

[14] González A., Pérez R., Verdegay J.L., *Learning The Structure of a Fuzzy Rule: a Genetic Approach* Proc. EUFIT'93 vol. 2 814-819 (1993). Also Fuzzy System and Artificial Intelligence, 3 n.1 57-70, (1994).

[15] González, A., Pérez, R., Valenzuela A., *Diagnosis of Myocardial Infarction through Fuzzy Learning Techniques.* Proc. IFSA'95 (1995).

[16] González, A., Pérez, R. *Completeness and Consistency Conditions for Learning Fuzzy Rules.* Technical Report #DECSAI-95103 (1995), also to appear in Fuzzy Sets and Systems.

[17] González, A., *A Learning Methodology in Uncertain and Imprecise Environments*, International Journal of Intelligent Systems, vol.19, 357-371, (1995).

[18] González, A., Pérez, R., *A Learning System of Fuzzy Control Rules.* In: F. Herrera, J.L. Verdegay (Eds.), Genetic Algorithms and Soft Computing, Physica-Verlag, 202-225, (1996).

[19] González, A. Herrera, F., *Multi-stage Genetic Fuzzy Systems Based on the Iterative Rule Learning Approach*, to appear in Mathware and Soft Computing (1996).

[20] Holland J.H., *Adaptation in Natural and Artificial Systems.* Ann Arbor: The University of Michigan Press. (1975).

[21] Holland J.H., Reitman, J.S., *Cognitive Systems Based on Adaptative Algorithms.* Pattern-Directed Inference Systems. New York, Academic Press, (1978).

[22] Hougen H.P., Villanueva, E. Valenzuela A., *Sudden Cardiac Death: a Comparative Study of Morphological, Histochemical and Biochemical Methods.* Forensic Science Internacional 52, pp. 161-169 (1992).

[23] Huang D., *The context-array Bucket Brigade Algorithm: An Enhanced Approach to Credit-Apportionment in Classifier Systems.* Proceedings of the Third International Conference on Genetic Algorithms. pp. 311-316.(1989).

[24] Isibuchi H., Morisawa T., Nakashima T., *Voting Schemes for Fuzzy-Rule-Based Classification Systems.* Proc. Fifth IEEE Intl. Conference on Fuzzy Systems,, pp. 614-620. (1996).

[25] Janikow C.Z., *A Knowledge-intensive Genetic Algorithm for Supervised Learning.* Machine Learning, 13, pp. 189-228. (1993).

[26] Magdalena L., Velasco J.R., *Fuzzy Rule-Based Controllers that Learn by Evolving their Knowledge Base.* In: F. Herrera, J.L. Verdegay (Eds.), Genetic Algorithms and Soft Computing, Physica-Verlag, 172-201, (1996).

[27] Michalski R.S., *Understanding the nature of learning*, Machine Learning: An artificial intelligence approach (Vol II). San Mateo, CA: Morgan Kaufmann (1986).

[28] Parodi, A., Bonelli, P., *A New Approach to Fuzzy Classifier System.* Proceedings of the Fifth International Conference on Genetic Algorithms, San Mateo, (1993).

[29] Quinlan J.R., *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers, San Francisco (1993).

[30] Sigillito V.G., Wing S.P., Hutton L.V., Baker K.B., *Classification of Radar Returns from the Ionosphere using Neural Networks.* Johns Hopkins APL Technical Digest 10, 262-266, (1989).

[31] Smith S.F., *A Learning System based on Genetic Algorithms.* Doctoral Dissertation, University of Pittsburgh, Departamente of Computer Science, (1980).

[32] Valenzuela-Redón, M., *The Fuzzy Classifier System: Motivations and First Results.* Parallel Problem Solving from Nature II, Springer Verlag , pp. 330-334 (1991).

[33] Venturini G., *SIA: a Supervised Inductive Algorithm with Genetic Search for Learning Attributes based Concepts*. Machine Learning: ECML-93, pp. 280-296 (1993).

[34] Weber R., *Fuzzy-ID3: A Class of Methods for Automatic Knowledge Acquisition* Proceedings of the 2nd International Conference on Fuzzy Logic and Neural Network, pp. 265-268 (1992).

[35] Zadeh, L.A., *The Concept of a Linguistic Variable and its Applications to Approximate Reasoning*. Part I Information Sciences 8, 199–249. Part II Information Sciences 8 301–357. Part III Information Sciences 9 43–80. (1975).