International Journal of Computational Intelligence Systems, Vol. 5, No. 2 (April, 2012), 212-230

An Efficient Inductive Genetic Learning Algorithm for Fuzzy Relational Rules

Antonio González, Raúl Pérez, Yoel Caises, Enrique Leyva

Departamento de Ciencias de la Computación e Inteligencia Artificial, Universidad de Granada, 18071-Granada, Spain

E-mail: {A.Gonzalez,Raul_Perez,ycaises,eleyvam}@decsai.ugr.es

Received 29 October 2010
Accepted 1 June 2011

Abstract

Fuzzy modelling research has traditionally focused on certain types of fuzzy rules. However, the use of alternative rule models could improve the ability of fuzzy systems to represent a specific problem. In this proposal, an extended fuzzy rule model, that can include relations between variables in the antecedent of rules is presented. Furthermore, a learning algorithm based on the iterative genetic approach which is able to represent the knowledge using this model is proposed as well. On the other hand, potential relations among initial variables imply an exponential growth in the feasible rule search space. Consequently, two filters for detecting relevant potential relations are added to the learning algorithm. These filters allows to decrease the search space complexity and increase the algorithm efficiency. Finally, we also present an experimental study to demonstrate the benefits of using fuzzy relational rules.

Keywords: Genetic fuzzy learning, fuzzy rules, fuzzy relational rules, classification.

1. Introduction

One of the most important areas in the development of fuzzy-logic-based applications has been, and remains, the design of fuzzy rule-based systems. Different types of fuzzy rules have been used for the different applications found in the literature, with the most common being the linguistic [1] and TSK rules [2]. The former is a traditional rule model whose antecedent and consequent are composed of linguistic variables, whereas the latter is based on rules whose antecedent is composed of linguistic variables and whose consequent is represented by a function of the input variables, frequently a linear function. The DNF rule is a very well known extension of the linguistic rule in which each input variable takes a set of linguistic terms whose members are joined by a disjunctive operator as its value, whilst the output variable remains a normal linguistic variable with a single associated value. Other types of fuzzy rule extensions, such as rules with weights, rules with double consequents, etc., each of which tries to improve the system accuracy and/or the system interpretability, have also been proposed.

The fuzzy relational rule (FRR) model has received relatively little attention to date [3,4]. The main idea underlying these type of rules and the mechanism to reason with them is introduced in [5]. FRRs include variables and fuzzy relations in the antecedent of the rule and generate a more flexible partitioning of the input space than traditional rules models. The main goal of this work is therefore to study whether this type of partitioning allows us to obtain more precise knowledge maintaining the simplicity in the description of a system, and checking if we can obtain fuzzy models with a good

interpretability-accuracy trade-off.

Despite their likely utility, very few learning methods have been proposed for FRRs. An inductive approach for learning FRRs was proposed in [3]. This algorithm was the extension of a fuzzy-rule-learning algorithm called SLAVE [6] and was able to manage FRRs which only consider general binary relations. Likewise, a fuzzy-rule-learning algorithm including particular binary relations on approximately linear dependence was proposed in [4].

Herein we propose, on the idea initiated in [3] and subsequently followed in [7], an efficient fuzzy rule learning algorithm that can handle the additional complexity that appears when we include fuzzy relations. To this end, and following the idea initiated in [3], we restrict the inclusion of fuzzy relations by using a catalog of relations that contains only those related to the problem. In [3] this catalog is previously defined by an expert and the learning algorithm is allowed to choose between the different relations included in the catalog. In this way, the catalog definition enables us to restrict the additional complexity involved in the learning of FRRs. However, definition of the catalog by an expert is not always the ideal solution since it can involve a considerable effort, therefore we now propose a methodology to automatically restrict the large number of fuzzy relations that can be considered and to select only the most promising for the problem at hand.

FRRs can be included using different kinds of learning algorithms, and in this work we use NSLV [8], a genetic fuzzy-rule-learning algorithm.

The following section is devoted to describing the FRR model and its inference process, the third section describes the most relevant details of the NSLV learning algorithm, and the modifications needed to allow this algorithm to extract FRRs, and the fourth section is devoted to describing how to manage fuzzy relations and the definition of a restrictive fuzzy relation selection process chiefly involving equality or comparison relationships. The fifth section is devoted to presenting an experimental study to analyze the possible interest of including relations in the learning process. Finally, the last section presents some conclusions and future work.

2. Fuzzy Relational Rules

Tradicionally, linguistic fuzzy models are based on families of fuzzy rules of the form:

$$\text{IF } X_1 \text{ is } A_1 \text{ and } X_2 \text{ is } A_2 \text{ and } \ldots \text{and } X_n \text{ is } A_n$$

$$\text{THEN } Y \text{ is } B$$

where $X_1, X_2, \ldots, X_n$ are the antecedent variables defined on the universes $U_1, U_2, \ldots, U_n$, and Y is the consequent variable defined on the universe V. $A_1, A_2, \ldots, A_n$ and B are fuzzy subsets defined on the respective universes. The basic effect of using this kind of rule can be seen as a rectangular partitioning of the input space (see Figure 1).

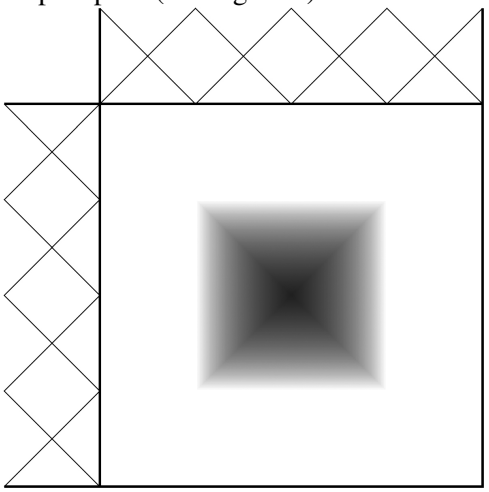

Fig. 1. Rectangular partition generated by a particular linguistic fuzzy rule associated with the central labels in the first and second variable

The use of fuzzy models with relational antecedents is introduced in [5]. An FRR is a rule in which the antecedent involves the satisfaction of a relationship between different variables, for example ,

$$\text{IF } (X_1, X_2, \ldots, X_n) \text{ is R THEN } Y \text{ is } B$$

where R is a fuzzy relation, in other words a fuzzy subset of $U = U_1 \times U_2 \times \ldots \times U_n$. An example of an FRR is

$$\text{IF } X_1 \text{ is approximately equal to } X_2 \text{ THEN } Y \text{ is } B.$$

Obviously, although this particular rule can be approximated by traditional fuzzy rules, the use of an FRR increases the interpretability of the knowledge

since it is described in a more human-like manner. Furthermore, it increases the simplicity of the model since only one rule is needed, whereas more rules (at least one for each fuzzy value of the variable) are needed in the previous model to represent the same knowledge.

A practical example of this type of rule in the design of a control rule for a mobile robot could be:

If the robot's distance to the right wall is **approximately equal to** *the distance from the robot to the left wall and the angle between the direction of the robot and the wall is almost zero then do not turn and increase speed.*

This rule can be represented satisfactorily by the FRR model using the particular relation "approximately equal to" (see Figure 2).

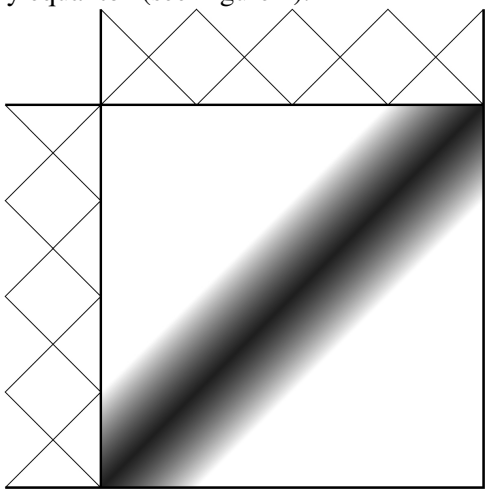

Fig. 2. Partition generated by the fuzzy relation *X is approximately equal to Y*

Thus, the idea is to include fuzzy relations in order to increase the knowledge-representation capacity of the learning system whilst improving the interpretability and simplicity of fuzzy models but extending the traditional fuzzy-rule model only minimally, in other words using fuzzy relations only when a clear improvement of the model results. Moreover, in a first step, we only explore a restrictive definition of the previous one in which we only consider antecedent requirements involving individual variables or the satisfaction of relationships between two variables, in other words we only consider binary relations such as

$$\text{IF } (X_1, X_2) \text{ is R and } X_3 \text{ is } A_3 \text{ THEN } Y \text{ is } B \qquad (1)$$

where R is now a fuzzy subset of $U = U_1 \times U_2$.

The use of FRRs allows more general partitions than the rectangular one, which, as can be seen from Figure 2, is obviously an extension of the traditional models.

An interesting study of the reasoning with these rules can be found in [9]. In the general case, this reasoning needs complex calculus, although the use of singletons as inputs simplifies the process. For an input like (x_1, x_2, x_3) with $x_i \in U_i$ for the rule described in (1), the output value is

$$B^*(y) = \tau \wedge B(y)$$

with $\tau = R(x_1, x_2) \wedge A_3(x_3)$ and $\wedge$ a t-norm like the minimum operator. When several rules are used, $B^*(y)$ is taken as the maximum value for the different $B^*(y)$ for each rule.

The aim of this work was to develop a FRR learning algorithm. However, as the number of possible binary relations between two selected variables among the set of n original variables can become too large and therefore cause problems for the search component of the learning algorithm, thereby increasing the complexity of the problem considerably, a previous reduction of the relation candidates is needed. In [3] we proposed a reduction method based on the definition of a catalog of fuzzy relations by an expert. The idea behind such a catalog is to collect those relations that the expert considers most relevant to the particular problem or to discard those that should not be used in any case.

The Catalog of Relations (CR) is actually managed as an index set. Let us suppose we number all the candidate relations, thus allowing us to define the Catalog of Relation CR as:

if $\{(X_i, X_j) \text{ is } R_k\}$ is a relevant relation to be consider for the learning algorithm, then we put (i,j,k) in the CR set.

We can now associate an integer representing the index of the relation in the catalog to this relation.

Using the CR set, the formalism for a FRR is

IF X_1 is $A_1 \wedge \ldots \wedge X_n$ is $A_n \wedge \{\wedge_{(i,j,k)\in H}[(X_i, X_j) \text{ is } R_k]\}$

THEN Y is B

where A_i are the antecedent values, B is the consequent value and $H \subseteq CR$ is an index subset defining the specific relation participating in the rule. We denote this rule as $R_B(A,H)$, with $A = (A_1, A_2, \ldots, A_n)$.

We will actually manage an extended version of this rule (the DNF rule) which allows us to assign a set of values from its domain to each antecedent variable of the rule, that is,

IF X_1 is $\tilde{A}_1 \wedge \ldots \wedge X_n$ is $\tilde{A}_n \wedge \{\wedge_{(i,j,k)\in H}[(X_i, X_J) \text{ is } R_k]\}$

THEN Y is B

where $\tilde{A}_i$ is a set of fuzzy values on universe U_i. The inference process associated to this rule is a simple extension of the general case. For an input $x = (x_1, x_2, \ldots, x_n)$, $x_i \in U_i$ the output is

$$B^*(y) = \tau(x, \tilde{A}_1, \ldots, \tilde{A}_n) \wedge R(x,H) \wedge B(y)$$

with

$$R(x,H) = \{\wedge_{(i,j,k)\in H} R_k(x_i, x_j)\} \quad (2)$$

and

$$\tau(x, \tilde{A}_1, \ldots, \tilde{A}_n) = \tilde{A}_1(x_1) \wedge \ldots \wedge \tilde{A}_n(x_n) \quad (3)$$

and

$$\tilde{A}_i(x_i) = \frac{max_{j\in S} A_{ij}(x_i)}{max_{j\in 1\ldots n_i} A_{ij}(x_i)} \quad (4)$$

where the domain of X_i is defined by n_i values $\{A_{i1}, A_{i2}, \ldots, A_{in_i}\}$ and $\tilde{A}_i$ is defined by the set of values contained in the index set S, in other words,

$$\tilde{A}_i = \{A_{ij} | j \in S\}.$$

Obviously, when $S = \{1, 2, \ldots, n_i\}$ then $\tilde{A}_i(x_i) = 1$ and the component i does not affect to the calculus of τ. The situation is therefore the same as when this variable does not appear in the rule.

Let us now see an example of this kind of rule. Suppose we have five variables X_1, X_2, X_3, X_4 and Y which take values on the same universe and with the same fuzzy domain associated for all the variables composed of the five fuzzy labels described in Figure 3.

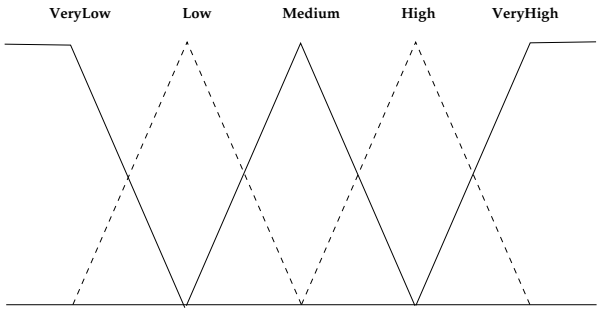

Fig. 3. Example of a fuzzy domain

An example of the particular type of rule we are using is:

IF X_1 is {VeryLow or Low} and
X_2 is {Medium or High or VeryHigh} and
X_3 is approximately equal to X_4
THEN
Y is Medium.

As we can see, the first variable takes the values VeryLow or Low, and using formula (4) this disjunction can be interpreted as the convex hull of both labels, in other words a new label means "less than or equal to Low". In this case, the set S used in equation (4) is S={1,2}. The second variable takes three values, and using the same formula can be interpreted as a new label meaning any value "greater than or equal to Medium". In this case, the set S used in equation (4) is S={3,4,5}.Finally, a relation associated with the third and fourth variable, specifically the relation "approximately equal to", appears.

Our rule model also considers a weight associated with each rule which represents the percentage of positive examples covered by the rule. We therefore adapted the inference mechanism to allow us to use FRRs with weights.

Now that we have described the FRR model, in the next section we will propose a learning algorithm that can be used to obtain this kind of rule.

3. Learning Algorithm

Our goal is to design a learning algorithm able to obtain a fuzzy relational rule set efficiently. This process could be made from scratch, but in our case we believe it is more efficient to change a previous proposed fuzzy rule learning algorithm for this purpose. In our case we extend NSLV [8], a learning algorithm of fuzzy rules. NSLV is the evolution of SLAVE [6], one of the first fuzzy rule learning algorithms.

3.1. The NSLV learning algorithm

NSLV is a genetic fuzzy rule learning algorithm that use a sequential covering strategy [10]. A prototypical description of this family of algorithms is:

SEQUENTIAL-COVERING(T,A,E,D)

- Learned-rules ← {}
- Rule ← LEARN-ONE-RULE(T,A,E)
- while PERFORMANCE(Rule,E) > D, do
  - Learned-rules ← Learned-rules + Rule
  - E ← E-examples correctly classified by Rule
  - Rule ← LEARN-ONE-RULE(T,A,E)
- Learned-rules ← sort Learned-rules accord to PERFORMANCE over E
- return Learned-rules

where T is the target attribute, A is the attribute set, E is the set of examples and D is a threshold.

The sequential covering algorithm reduces the problem of learning a disjunctive set of rules to a sequence of simpler problems, each of which requires a single conjunctive rule to be learned.

The algorithm NLSV [8] does not actually delete the examples correctly classified by a fuzzy rule, as shown in the above description, but instead marks the examples already covered by this rule with the idea that they should not be considered as positive examples for other rules, but can be considered as negative. Moreover, NLSV uses a genetic algorithm (GA) to implement the LEARN-ONE-RULE procedure. The input of this GA is a target attribute representing the consequent variable, the complete set of antecedent variables and the set of examples, and the output is a single rule that covers at least some of the examples. The criterion for selecting the best rule in NSLV is based on obtaining a single rule that verifies the following objectives:

O1: The rule describes a high number of examples correctly and a low number of them incorrectly.

O2: The rule has a reduced number of variables in its antecedent.

O3: The rule antecedent has an understandable assignment of values to variables.

The criterion for selecting the best rule is therefore actually managed as a multiobjective problem since the solution must satisfy three different criteria. To solve this problem, NSLV applies a lexicographical evaluation functional (LEF), in other words it assumes that the objectives do not have the same importance. The preference criterion in NSLV corresponds exactly to the order in which the objectives have been written previously.

Thus, the most important criterion (**O1**) involves finding the rule that covers many of the positive examples and few of the negative examples. In classical learning algorithms this criterion is related to two well-known conditions, namely the consistency and completeness [11]. These conditions are extended for working with fuzzy rules in [6]. Such an extension establishes the degree to which a rule satisfies each of them, thus meaning that NSLV tries to optimize the product of consistency and completeness. Both definitions require the calculus of the number of positive and negative examples to a fuzzy rule. Given a particular fuzzy rule

$$\text{IF } X_1 \text{ is } \tilde{A}_1 \wedge \ldots \wedge X_n \text{ is } \tilde{A}_n \text{ THEN } Y \text{ is } B$$

the number of positive examples of this rule is defined as the cardinal of the fuzzy set of positive examples to the rule. This set is defined by giving a membership degree to each example regarding the concept of ”being positive” to the rule. This mem-

bership degree is defined by

$$\tau(x, \tilde{A}_1, \ldots, \tilde{A}_n) \wedge \frac{B(y)}{max_{B'} B'(y)}$$

and represents the simultaneous adaptation of the example to the antecedent and the consequent of the rule. The number of negative examples is the cardinal of the fuzzy set of the negative examples to the rule. The membership degree of each example to this set is defined by

$$\tau(x, \tilde{A}_1, \ldots, \tilde{A}_n) \wedge \frac{max_{B' \neq B} B'(y)}{max_{B'} B'(y)}$$

and represents the adaptation to the antecedent and any of the other possible values of the consequent variable different to that considered in the rule.

The other two objectives (**O2** and **O3**) are related to the simplicity of the rule. The second objective establishes a preference for rules with a reduced number of variables. An example of the application of the third objective is the preference for the assignment of values that can be replaced by new labels such as *"the variable is greater than"* or *"the variable is less than"*, such as those described in the example provided in Section 2.

Each individual in the population of the genetic algorithm represents a complete rule. NSLV uses a DNF-type rule model, therefore each variable takes as its value a set of linguistic terms whose members are joined by a disjunctive operator. The genetic algorithm represents each DNF rule by three different elements or levels (Figure 4):

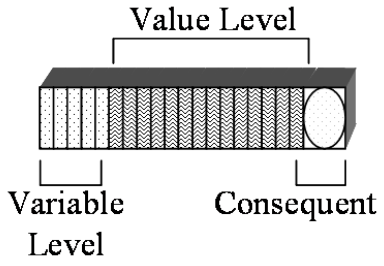

Fig. 4. Representation of a rule in NSLV

- *The variable level*. This level contains a gene for each predictive variable involved in the problem. Each gene represents a real value that it is interpreted as the relevance degree of a predictive variable on the rule in relation to the consequent variable. Furthermore, a special gene is added in this level that it is interpreted as an activation threshold, in other words the variables whose associated genes have values less than the threshold are not considered in the antecedent of the rule. The use of this level therefore allows us to develop an embedded feature-selection process [12].
  Two genetic operators are used in this level: the two-point crossover and the real uniform mutation. Generation of this level in the initial population is based on an information measure obtained from the training set. This measure permits an initial relevance degree to be established for each variable. The values obtained are used for all individuals in the population on this level. The activation threshold is randomly generated between the maximum and minimum value of the initial relevance degree obtained. A more detailed description can be found in [13].
- *The value level*. This is composed of the sequence of assignments to the predictive variables, where each assignment variable/value is represented by a binary string. The complete level is composed by concatenation of the binary strings representing the assignment variable/values for all input variables. The assignment of a certain variable will appear finally in the description of the rule if the value of its associated gene in the variable level permits it to be considered as the relevant variable for this rule.
  We use a binary representation and two genetic operators on this level: the two-point crossover and the binary uniform mutation. Generation of this level in the initial population is as follows: as the consequent value is known (generated previously in the consequent level), an example of this class is selected. The more specific antecedent for this example is obtained and is used to code the individual. This process is repeated for all indi-

viduals in the population.

- *The consequent level*. This codes the value of the classification variable for the rule. This level is composed of one gene which is represented by an integer value.
  The integer uniform mutation is the only genetic operator considered on this level, and this level is generated randomly in the initial population.

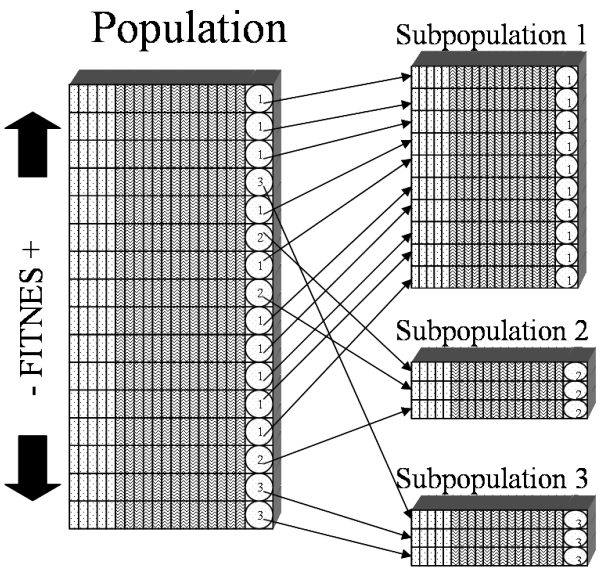

Fig. 5. Subpopulations in NLSV

NSLV uses a steady-state genetic algorithm that divides the population into subpopulations, each containing the best individuals per class (Figure 5). It uses the classical reproduction process of the steady-state genetic algorithm but with one modification: the worst individuals to be removed from the population are selected in those subpopulations that have more than a minimum number of individuals, thus guaranteeing that the subpopulations have sufficient individuals to evolve during the genetic search. The selection mechanisms choose two individuals, which generate two offspring to replace those individuals selected for removal.

The goal of the subpopulations is to keep high diversity in the genetic population, thus allowing premature convergence of the search process to be controlled. Furthermore, the sizes of the subpopulations are not fixed as the size depends on the search state. The policy followed in NSLV consists of prioritising those subpopulations which present a high capacity to improvise their individual members. Two indicators are taken into account when evaluating this capacity:

(a) the difference between the fitness of the best individual in the subpopulation and the number of examples of its class, and
(b) the difference between the fitness of the best and the worst individuals.

In both cases, the subpopulation is close to a good point of convergence when the differences are low and therefore the subpopulation does not need to contain a large number of individuals. In contrast, when the subpopulation is not near an acceptable point of convergence the subpopulations need to contain more individuals to evolve.

The minimum size of the subpopulation for a particular class ($MinSize(class)$) is defined as

$\frac{SizePopulation}{2 \times NumberClasses}$ if the subpopulation is near of an acceptable convergence point.

$\frac{SizePopulation}{1+NumberClasses}$ in other case.

where $SizePopulation$ is the number of individuals in the genetic population and $NumberClasses$ is the number of classes of the learning problem.

An exception to the previous policy is applied when a determined class contains no examples. In this case, the subpopulation associated with this class is removed from the genetic population.

3.2. NSLV-R: an FRR genetic-learning algorithm

In this section we will adapt NSLV so that it can learn fuzzy relational rules; the new algorithm is called NSLV-R.

One important aspect in this adaptation is obviously the modification of the genetic algorithm, particularly the modification of the representation of a rule. The first component we need to modify is the genetic representation of each rule. Thus, a FRR is represented by extending the representation of a previously described fuzzy rule. To this end, we again consider the variable level, the value level and the consequent level with the same interpretation, structure and operators, but we now include a new substructure to encode the active fuzzy relations that appear in the rule. This structure will be called the relation level.

The relation level. This level represents the relation set included in the antecedent of the rule. Each gene codes a possible relation of the Catalog of Relations.

Each individual is now composed by four different representations or levels (Figure 6).

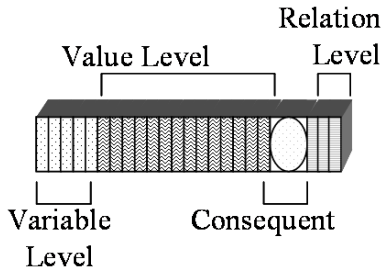

Fig. 6. Representation of a rule in NLSV-R

The maximum number of relations that we can include in a rule is defined by a parameter. In the experiments we have considered up to 10 relations in each rule. The relation level uses integer coding, where 0 indicates "no relation" and any other positive value means to include the relation associated with this index in the Catalog of Relations.

As far as genetic operators are concerned, we keep the same operators used by NSLV in the description part associated with variables, antecedent values and consequent value, whereas in the relation level we use a crossover operator which corresponds to the integer coding extension to the intersection of two points used in binary coding. We also use in the relation level an operator of mutation associated with two probabilities, one associated with the probability of applying the operator (we used a value of 0.99 in the experiments) and the other with a non-uniform probability distribution that gives a higher probability of zero than the remaining values. The idea is to include relations only when there is a clear improvement in the accuracy of the rule.

In the initial population, all the genes for all individuals of this level begin with a value of zero, in other words no relations are considered in the chromosome at the beginning of the genetic process.

Finally, the criterion for selecting the best rule is similar to NSLV and is based on the multiobjective process defined in subsection 3.1. However, some aspects need to be adapted. Thus, in relation to objective **O1**, it is necessary to clarify the calculus of the number of positive and negative examples for a FRR. This calculation is needed for the consistency and completeness definitions, which must be adapted to the new structure of the rule.

Given a FRR $R_B(\tilde{\mathrm{A}}, H)$, the number of positive examples of this rule is defined in a similar manner as for fuzzy rules without relations except that the membership degree to each example of "being positive" to the rule is now defined by

$$\tau(x, \tilde{\mathrm{A}}_1, \ldots, \tilde{\mathrm{A}}_n) \wedge R(x, H) \wedge \frac{B(y)}{max_{B'} B'(y)}$$

using the τ and R definitions described in equation (2) and (3). The previous expression represents the simultaneous adaptation of the example to the antecedent (variables and relations) and the consequent of the rule. The membership degree of each example of "being negative" to the rule is defined by

$$\tau(x, \tilde{\mathrm{A}}_1, \ldots, \tilde{\mathrm{A}}_n) \wedge R(x, H) \wedge \frac{max_{B' \neq B} B'(y)}{max_{B'} B'(y)}$$

which represents the adaptation to the antecedent (variables and relations) and any of the other possible values of the consequent variable different to that considered in the rule.

In relation to the objective **O2**, we include the sum of the number of variables and the number of

relations in the comparison in order to obtain rules with the smallest number of variables and relations. The objective **O3** is not affected by the inclusion of relations.

4. The Catalog of Relations

The aim of this section is to establish a mechanism to limit the number of relations in the catalog. In any case, this mechanism is only associated with the comparison or equality relations established for fuzzy variables defined on a real valued universe.

Our proposal is to define two filters for the possible relations. In the first filter, relations are eliminated through an approach based on the properties of the variables or their reference universes. The second filter uses a heuristic procedure based on the use of information measures.

Before that, however, we describe the relations that we use in our study and the procedure used to define them.

4.1. Relations

Herein we have considered only four fuzzy relations and two crisp relations, although of course we could consider any other relation that may be of interest. The reason for choosing these particular relations is mainly for simplicity. With the idea to check that relations could be of interest to a broad list of databases, such as those we use in the experimental section, it is advisable to select a set of simple relations that are likely to be useful for the different databases under consideration.

In particular, we have considered the fuzzy relations:

- approximately equal to,
- very different to
- approximately less than or equal to, and
- approximately greater than or equal to

and the two crisp relations $>$ and $<$.

All these relations are defined for numerical variables X_i and X_j with bounded universes $[inf_i, sup_i]$ and $[inf_j, sup_j]$, respectively.

Furthermore, all the fuzzy relations have been defined using a particular methodology with the aim of transforming the fuzzy binary relation into a variable taking values in a particular universe, in other words to transform $R(X_i, X_j)$ in the variable Z_{ij} taking a particular fuzzy value. For example, X_1 *is approximately equal to* X_2 is transformed into $Z_{ij} = |X_i - X_j|$ *is approximately equal to zero*, where zero is a fuzzy value.

Using this methodology, we can define the different fuzzy relations in a simple manner. Thus, associated with the universe of the variables X_i and X_j we define the parameter

$$q_{ij} = \frac{|I_i - I_j|}{c}$$

with $I_i = min\{sup_i, sup_j\}$, $I_j = max\{inf_i, inf_j\}$ and where c is a strictly positive parameter (the experiments were performed using c=10). The value q_{ij} represents the size of the partition in c fragments of the universe of the difference variable $|X_i - X_j|$ as shown in Figure 7.

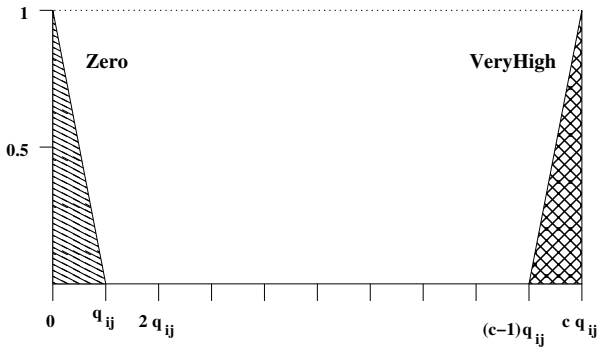

Fig. 7. Relations *approximately equal to* and *very different to*.

The X_i is *approximately equal to* X_j relation is associated with $|X_i - X_j|$ is *approximately equal to Zero*. Zero is defined as the fuzzy number described in Figure 7 and therefore calculated as:

$$\mu_{X_i \approx X_j}(x_i, x_j) = \begin{cases} 1 - |\frac{x_i - x_j}{q_{ij}}| & \text{if } |x_i - x_j| < q_{ij} \\ 0 & \text{otherwise} \end{cases}$$

The X_i is *very different to* X_j relation is associated with $|X_i - X_j|$ is *approximately equal to VeryHigh*, where VeryHigh is defined as the fuzzy number described in Figure 7 and therefore calculated as:

$$\mu_{X_i <> X_j}(x_i, x_j) = \\ = \begin{cases} 0 & \text{if } |x_i - x_j| < (c-1)q_{ij} \\ \frac{|x_i - x_j| - (c-1)q_{ij}}{q_{ij}} & \text{otherwise} \end{cases}$$

The X_i is *approximately less than or equal to* X_j relation is a combination of the relation $<$ and the *approximately equal to* relation:

$$\mu_{X_i < \approx X_j}(x_i, x_j) = \\ = \begin{cases} 1 & \text{if } x_i < x_j \\ 1 - |\frac{x_i - x_j}{q_{ij}}| & \text{if } 0 < |x_i - x_j| < q_{ij} \\ 0 & \text{otherwise} \end{cases}$$

The X_i is *approximately greater than or equal to* X_j relation is included in an indirect manner since $X_i > \approx X_j$ is equivalent to $X_j < \approx X_i$.

4.2. Basic filter

The main idea of the first filter is to use expert information or information generated by its own variables to reduce the number of relations that the learning algorithm can include in the rules. The idea now is to consider the following three processes to reduce the number of relations:

- The first idea is to use expert information, if available. This information can be very useful and is easily described in some cases. For example, *Do not use relations comparing age and height*.
- The second idea is to use information regarding the units of the variables. Thus, only comparison relations between variables with the same units are allowed.
- Finally, we only consider comparison relations between variables with a sufficient overlap of their universes.

The first two processes are simple and require no additional explanation, although further details regarding how to make the third are required. Thus, the idea is to define a degree of overlap between variables such that if this degree of overlap exceeds a certain threshold, then comparison or equality relations between both variables are candidates to be included in the catalog of relations (CR).

Given the variables X_1 and X_2 with universes $[inf_1, sup_1]$ and $[inf_2, sup_2]$ respectively, we can say that there is overlap between them if

$$(inf_2 \leqslant inf_1 \leqslant sup_2) \; or \; (inf_1 \leqslant inf_2 \leqslant sup_1).$$

If there is no overlap between these two variables, then the degree of overlap is zero. In contrast, we can define the degree of overlap of variables X_1 and X_2 as

$$degree_o(X_1, X_2) = max(\frac{|b-a|}{|sup_1 - inf_1|}, \frac{|b-a|}{|sup_2 - inf_2|}),$$

where $a = max(inf_1, inf_2)$ and $b = min(sup_1, sup_2)$.

When defining the CR we usually consider a degree of overlap close to one, for example 0.9. Thus, if $degree_o(X_i, X_j) > 0.9$ then relations between both variables are allowed by the basic filter.

4.3. Heuristic filter

The second filter reduces the number of relations to be included in the CR using heuristic information. The idea is to use a measure of information to select the relations that provide more information regarding the variable consequence. This is clearly a heuristic process since we use only partial information about each relation, which means that it is not possible to determine the influence of the relation in a specific rule exactly due to the existence of other variables and/or relations.

The heuristic filter is used on the relations obtained after applying the basic filter described previously. Thus, the information measure is applied to each element $\{(X_i, X_j) \text{ is } R_k\}$ obtained in the previous stage. This information measure is

actually applied to the variable associated with the fuzzy relation $Z_{ij} = |X_i - X_j|$ using the following idea: the information measure between the relation $\{(X_i, X_j)$ is $R_k\}$ and the classification variable will be calculated using the information measure between $Z_{ij} = |X_i - X_j|$ is V_k and the classification variable, where V_k is a particular value that defines the fuzzy relation (zero in the case of the relation "approximately equal to").

The idea is to use a measure of the relevance of each variable associated to the relation with respect to the classification variable.

We use the following measure [14,15]

$$\rho(X,Y) = \frac{I(X,Y)}{H(X,Y)}$$

where

$$I(X,Y) = \sum_x \sum_y -p(x,y)\ log_2(\frac{p(x)p(y)}{p(x,y)})$$

and H(X,Y) is the Shannon entropy over two variables, defined as

$$H(X,Y) = \sum_x \sum_y -p(x,y)\ log_2\ p(x,y).$$

The ρ measure estimates the dependence between two generic variables X and Y in the following way: values of $\rho(X,Y)$ close to zero determine a high degree of independence of both variables, whereas values close to one demonstrate a high degree of functional dependency between them.

In our case, we calculate

$$\rho(Z_{ij},Y) = \frac{I(Z_{ij},Y)}{H(Z_{ij},Y)}$$

where Z_{ij} is the variable associated with the relation $\{(X_i, X_j)$ is $R_k\}$ and Y is the classification measure.

Thus, the procedure consists in selecting the N_R best relations, where N_R is a parameter that we need to fix. In order to select these relations, we first use $\rho(Z_{ij},Y)$ to order the relations and then select the best relations amongst these that verify the following property on the basis of this ordering

$$\rho(Z_{ij},Y) > max_p\{\rho(X_p,Y)\}$$

where X_p are the different antecedent variables and Y is the classification variable. The previous expression ensures that the information regarding the relation is better than the information provided by any of the antecedent variables.

Moreover, in this selection we include useful relations for the different values of the consequent variable.

Table 1. Database used in the experimental study. Each row represents the number of examples (#E), variables (#V), continuous variables (#C), nominal variables (#N), classes (#Cl) and missing values (M).

Database	#E	#V	#C	#N	#Cl	M
ann	898	38	6	32	5	No
aut	205	25	15	10	6	Yes
brd	106	11	4	7	7	No
bal	625	4	4	0	3	No
bpa	345	6	6	0	2	No
car	1729	6	0	6	4	No
crx	690	15	6	9	2	No
col	368	22	7	15	2	Yes
drm	366	34	33	1	7	No
gls	214	9	9	0	6	No
h-c	303	13	6	7	2	Yes
h-h	296	13	6	7	5	No
h-s	270	13	13	0	2	No
hpt	155	19	6	13	2	Yes
irs	150	4	4	0	3	No
pim	768	8	7	1	2	No
son	208	60	60	0	2	No
tao	888	2	2	0	2	No
thy	215	5	5	0	3	No
trf	748	4	4	0	2	No
vot	435	16	0	16	2	No
wdbc	569	30	30	0	2	No
wpbc	198	33	33	0	2	Yes
zoo	101	17	0	17	7	No

5. Experimental Study

In this section we compare the behaviour of the new NSLV-R learning algorithm with that of the base algorithm NSLV. Furthermore, NSLV-R is benchmarked against other well-known classical and fuzzy-rule-learning algorithms. The experimen-

Table 2. Results obtained using NSLV and NSLV-R

	Training		Test		Rules		Time		Conditions	
Data	NSLV	NSLV-R	NSLV	NSLV-R	NSLV	NSLV-R	NSLV	NSLV-R	NSLV	NSLV-R
ann	94.8 (2)	96.3 (1)	93.7 (2)	95.1 (1)	8 (1)	9 (2)	102 (1)	128.7 (2)	15.3 (1)	16.8 (2)
aut	92.5 (2)	94.4 (1)	69.8 (2)	70.7 (1)	17.6 (1.5)	17.6 (1.5)	78 (1)	100.7 (2)	83.2 (2)	65.5 (1)
bal	86.6 (2)	88.4 (1)	77.9 (2)	80.2 (1)	18.8 (2)	18.1 (1)	49 (1)	55.1 (2)	43.5 (1)	44.7 (2)
bpa	66.9 (2)	75.9 (1)	56.7 (2)	71 (1)	6 (1)	7.2 (2)	18 (2)	15 (1)	18 (2)	16 (1)
brd	94.7 (1)	93.5 (2)	55.6 (2)	65 (1)	19.1 (2)	18 (1)	17 (1)	27.6 (2)	47.7 (2)	42.5 (1)
car	86.4 (2)	88.2 (1)	85.5 (2)	87.5 (1)	14.5 (1)	17.1 (2)	77 (1)	123.7 (2)	34.2 (1)	39.6 (2)
col	91.4 (1)	90.5 (2)	82 (2)	84.2 (1)	7.7 (2)	6.7 (1)	48 (2)	42.5 (1)	25.1 (2)	20.8 (1)
crx	90.2 (2)	91 (1)	83.7 (2)	83.9 (1)	8 (1)	9 (2)	40 (1)	69.3 (2)	27.1 (1)	31.5 (2)
drm	99.2 (1)	99 (2)	94.7 (2)	95.6 (1)	9.5 (1)	10.3 (2)	58 (1)	145 (2)	22.8 (2)	20.8 (1)
gls	80.7 (2)	72.6 (1)	63.4 (1)	57.3 (2)	14.5 (2)	12.3 (1)	40 (2)	35.9 (1)	41.5 (2)	35 (1)
h-c	91.2 (2)	92.7 (1)	77.8 (1)	75.2 (2)	11.4 (1)	13.5 (2)	33 (1)	43.5 (2)	36.8 (1)	44.3 (2)
h-h	87.8 (2)	88.7 (1)	66.3 (1)	66.2 (2)	37.6 (2)	34 (1)	74 (1)	113.4 (2)	120.5 (2)	111.2 (1)
hpt	91.4 (2)	93.1 (1)	79.9 (2)	85.7 (1)	5 (1)	6.7 (2)	20 (2)	9.7 (1)	14.5 (1)	16 (2)
h-s	91.5 (2)	92 (1)	77.4 (1)	72.9 (2)	10.8 (2)	10.3 (1)	34 (2)	33 (1)	32.7 (2)	31.8 (1)
irs	96.8 (2)	97.1 (1)	93.9 (1.5)	93.9 (1.5)	4 (1.5)	4 (1.5)	3 (2)	2 (1)	3.5 (1)	4.2 (2)
pim	80.2 (1.5)	80.2 (1.5)	72.6 (2)	73.8 (1)	12.8 (2)	11 (1)	87 (2)	84 (1)	39 (2)	33.2 (1)
son	90.4 (1)	90.1 (2)	69.1 (2)	72.9 (1)	9.9 (2)	9.6 (1)	60 (1)	87 (2)	48 (2)	41.7 (1)
tao	82.4 (2)	82.7 (1)	81.6 (2)	82.3 (1)	3.2 (1)	4 (2)	22 (1)	25 (2)	5 (1)	6.7 (2)
thy	93.5 (2)	94.7 (1)	90.7 (2)	92.6 (1)	4.9 (2)	4.7 (1)	8 (2)	6 (1)	10.5 (1)	10.6 (2)
trf	77.1 (2)	77.4 (1)	75.9 (1.5)	76.4 (1.5)	3.2 (2)	2.8 (1)	7 (1)	11 (2)	4.5 (2)	4.2 (1)
vot	97.6 (2)	97.7 (1)	95.6 (2)	96.5 (1)	4.7 (1.5)	4.7 (1.5)	7 (1)	11 (2)	9.6 (2)	9.5 (1)
wdbc	95.8 (1)	94.2 (2)	93.6 (1)	92.2 (2)	4.5 (1)	5.2 (2)	75 (2)	45 (1)	17.5 (2)	15.4 (1)
wpbc	80.1 (2)	81.2 (1)	74.2 (1)	71.2 (2)	3.2 (1)	4.2 (2)	15 (2)	11 (1)	9.6 (2)	8.6 (1)
zoo	99.5 (2)	100 (1)	95.8 (2)	96.4 (1)	7.4 (1)	7.5 (2)	6 (1)	7 (2)	9.4 (1.5)	9.4 (1.5)
Rnk	1.73	1.27	1.73	1.27	1.48	1.52	1.42	1.58	1.6	1.4
Pos	2	1	2	1	1	2	1	2	2	1

tal study was performed on 24 databases (Table 1) from the UCI Repository of Machine Learning Databases and Domain Theories [16].

We have used five uniformly distributed linguistic labels to define the domain of the continuous variables in all databases. In both algorithms, we use the missing-values treatment described in [17]. The results were obtained by ten-fold cross validation using the same partition for all algorithms. The size of the population was fixed at 100. The GA finishes returned a solution when the best individual did not change in the population during 500 iterations.

The variable-level mutation probability is

$$\frac{1}{\#V}$$

where $\#V$ is the number of predictive variables involved in the learning problem.

The value-level mutation probability is

$$\frac{1}{Length\_Chromo}$$

where *Length_Chromo* is the number of bits needed to code the chromosome level of an individual. We focused our study on the following indicators: accuracy on training set (Training), accuracy on test set (Test), the number of rules (Rules), the time needed to learn the whole model (Time), and, finally, the number of conditions of the whole rule set (Conditions). For the NSLV-R algorithm, this last parameter is evaluated as the number of variables plus the number of relations.

Training and Test represent the accuracy of the learned model, whereas Rules and Conditions define the size and interpretability of the knowledge base and Time shows the learning efficiency.

A Bonferroni-Dunn test [18] has been used, which establishes that two classifiers have differences if they differ in at least one critical distance, was used to perform the comparison. This distance depends on the number of databases, the number of learning algorithms to be compared and a critical value q_α, with error α. Thus, setting α to a value 0.05 gives $q_\alpha = 1.960$, therefore the critical distance for 24 databases and two classifiers is **0.4**.

The results obtained are shown in Table 2, where

the first column represents the results obtained using NSLV and the second column those obtained using NSLV-R. The first row lists the indicators evaluated, and the last but one shows the values used for the Bonferroni-Dunn test.

It is clear from Table 2 that there are significant differences in terms of Training and Test, thus suggesting that NSLV-R is more accurate than NSLV. The improvement in Training indicates that the inclusion of relations provides the learning algorithm with a greater ability to represent the underlying knowledge of the problem, whereas the improvement in Test validates the improvement in training in the sense that it is not due to overlearning.

Table 3. Rule Set obtained for the IRIS Database using NSLV-R

IF Petal_length $<\approx$ Sepall_width THEN Class is Setosa WITH weight 1.0
IF Petall_width IS High THEN Class IS Virginica WITH weight 0.97
OTHERWISE Class is Versicolour WITH weight 0.33

No significant differences were observed between the two algorithms in terms of the parameters Rules and Conditions, although NSLV-R tended to increase the number of rules and decrease the number of conditions. Thus, the number of conditions per rule in NSLV is in the interval [0.875,4.85], with an average of 2.7, in this study. This value is low considering that the average number of variables involved in the selected database is 16.96, thus meaning that the individual rules have a reduced number of conditions in their antecedents. The number of conditions obtained by NSLV-R is in the range [1.05,4.34], with an average of 2.5. A comparison of these results shows that NSLV-R produces simpler individual rules. A rule set obtained by NSLV-R for the IRIS database is shown in Table 3. The knowledge is composed by a reduced set of simple rules which use the *approximately less than or equal to* relation.

Table 4. Relations included in each database. #R represents the average number of relations in each rule set and %R the average ratio of relations with respect to the total number of conditions in each rule set

DBase	#R	%R	DBase	#R	%R
ann	1.6	10%	h-s	4.4	27%
aut	5.5	8%	hpt	2.3	7%
bal	0	0%	irs	1.3	31%
bpa	4.4	27%	pim	1.0	3%
brd	0.5	1%	son	6.7	16%
car	0	0%	tao	0	0%
col	0.2	0.1%	thy	3.7	35%
crx	4	13%	trf	0.5	12%
drm	5.5	26%	vot	0	0%
gls	7.7	22%	wdbc	5.5	36%
h-c	5.2	12%	wpbc	3.7	43%
h-h	7.5	6%	zoo	0	0%

As mentioned above, the algorithm includes relations only when needed to improve the accuracy. We can see this effect by looking at Table 4, which shows the average number of relations used and the ratio of conditions that are relations on each database.

Analysis of the computing time is also interesting. Thus, although there are no significant differences in time, given that the inclusion of relations increases the search space and that, furthermore, the new algorithm takes time to build the CR set, maintaining the computing time can be interpreted positively. The time required to build the CR is justified as this process reduces the time needed to converge to good solutions and we know from the significant improvements in test and training that the solutions obtained are better than those provided by the previous version.

5.1. Comparing NSLV-R with other classical learning algorithms in terms of accuracy

In this section we compare NSLV-R with some other classification algorithms using the WEKA platform [19] and the default configuration for each algorithm.

In order to find significant differences among the methods, we use a statistical analysis. According to the recommendations made in [20], where the use of

Table 5. Test accuracy for NSLV-R and several well-know classical learning algorithms

Databases	C4.5	IB5	NBayes	Part	SMOpol	NSLV-R
ann	98.9 (1)	97.3 (2)	86.3 (6)	98.6 (2)	91.9 (5)	95.1 (4)
aut	80.9 (1)	64 (4)	58.1 (5)	74.4 (2)	45.6 (6)	70.7 (3)
bal	77.4 (6)	88.2 (3)	90.6 (1)	82.9 (4)	88.3 (2)	80.2 (5)
bpa	62.3 (3)	58.9 (4)	56 (6)	67.6 (2)	58 (5)	71 (1)
brd	68.8 (1)	57.5 (6)	67.9 (5)	64.1 (2)	65 (3.5)	65 (3.5)
car	92.4 (4)	95.3 (2)	85.7 (6)	95.8 (1)	93.2 (3)	87.5 (5)
col	85.3 (1)	81.5 (5)	78.2 (6)	84.5 (2)	82.4 (4)	84.2 (3)
crx	83.7 (3)	81.4 (5)	76.9 (6)	83.3 (4)	84.6 (1)	83.9 (2)
drm	96.1 (3)	95.3 (5)	97.8 (1)	94.2 (6)	97.2 (2)	95.6 (4)
gls	66.1 (2)	64.7 (3)	48.9 (5)	66.6 (1)	35.6 (6)	57.3 (4)
h-c	78.4 (4)	83.2 (1)	82.8 (2)	74.2 (6)	82.5 (3)	75.2 (5)
h-h	66.6 (2.5)	62.2 (6)	64.6 (5)	67 (1)	66.6 (2.5)	66.2 (4)
h-s	79.3 (5)	80.7 (3)	83.3 (1)	80 (4)	82.6 (2)	72.9 (6)
hpt	79.3 (6)	81.9 (5)	83.8 (3.5)	85.8 (1)	83.8 (3.5)	85.7 (2)
irs	94 (5)	96 (2)	96 (2)	94 (5)	96 (2)	94 (5)
pim	74.2 (3)	73.3 (5)	75.8 (2)	78.9 (1)	65.1 (6)	73.8 (4)
son	71 (4)	84 (1)	69.7 (5)	74.4 (2)	69.3 (6)	72.9 (3)
tao	95.9 (2)	97.1 (1)	81 (6)	94.3 (3)	83.6 (4)	82.3 (5)
thy	85.5 (6)	88.8 (3)	87.0 (5)	96.3 (1)	88.5 (4)	92.6 (2)
trf	78.3 (1)	68.8 (6)	75.1 (5)	77.9 (2)	76.3 (4)	76.4 (3)
vot	96.7 (1)	91.4 (5)	90.3 (6)	96.3 (3)	95.6 (4)	96.5 (2)
wdbc	94.4 (3)	96.8 (1)	93.1 (4)	94.5 (2)	92.9 (5)	92.2 (6)
wpbc	71.6 (3)	78.8 (1)	69.4 (6)	70 (5)	73 (2)	71.2 (4)
zoo	92.8 (4)	90.5 (6)	94.5 (2)	93.8 (3)	76 (6)	96.4 (1)
Average	82.1	81.6	78.9	82.9	78.1	80.8

a set of simple, safe and robust non-parametric tests for statistical comparisons of classifiers has been introduced. Specifically, we use the Friedman's [21] and Holm's [20] tests. In order to perform multiple comparisons, it is necessary to check whether all the results obtained by the algorithms present any significant difference (Friedman's test) and, in the case of finding one, then we can find out by using a post-hoc test for comparing the control algorithm with the remaining algorithms (Holm's test). We use $\alpha = 0.05$ as the level of confidence in all cases.

The algorithms considered for this study are:

- C4.5 [22], which is based on decision trees and it is an extension of the ID3 algorithm for working with continuous variables and missing values.
- IBk [23], which is based on the Nearest Neighbour Algorithm and classifies an instance with the majority class of its k nearest neighbours. We set the k parameter to 5.
- NBayes [24], which is a probabilistic classifier that estimates parameters in a Bayesian model.
- Part [25], which is a learning framework that applies divide and conquer learning techniques to rules for developing a classifier.
- SMOpol [26], which is an implementation of support vector machines. In this study we used SMO with a polynomial (order 3) kernel.

This list contains several well-known algorithms, each of which represents a different learning paradigm. The test accuracy for these algorithms is shown in Table 5 where the last rows of this table show the average accuracy obtained by each algorithm on all databases.

Table 6. Friedman statistic (distributed according to chi-square with 5 degrees of freedom)

Statistic	critical value	p value
7.751905	11.0705	0.169855

Table 7. Average rankings of the algorithms (Friedman)

Algorithm	Ranking
Part	2.8125
C45	3.0833
IB5	3.5417
NSLV-R	3.6458
SMOpol	3.8125
Nbayes	4.1042

Table 6 shows the result of the Friedman's test with α = 0.05. From this result, we can conclude that here are not significant differences between the studied methods since the statistic is not greater than the critical value. Although this test shows no difference between methods, in order to confirm the results we apply the Holm's test. Average ranks obtained by each method in the Friedman's test are shown in Table 7. These averages are calculated by the Friedman's test, and establish an order of preference which is used by the Holm's test.

Table 8. Holm's test comparison Table for $\alpha = 0.05$

i	algorithm	z	p	α/i
5	Nbayes	2.391702	0.01677	0.01
4	SMOpol	1.85164	0.064078	0.0125
3	NSLV-R	1.543033	0.122823	0.016667
2	IB5	1.350154	0.176966	0.025
1	C45	0.501486	0.616029	0.05

We now apply Holm's test to compare the best ranking method (Part) with the remaining methods. Table 8 presents these results. In this Table, the methods are ordered with respect to the z value obtained. Holm's test rejects the hypothesis of equality with the rest of the methods ($p < \alpha/i$). In this case, we can see that the hypothesis is accepted in all cases and therefore, there are not significant differences between the methods.

So, we can interpret that NSLV-R has a prediction capacity similar to the classical studied algorithms. Figure 8 shows the expected accuracy of each classifier for the databases studied here. *P*art is the best algorithm as regards this parameter, with NSLV-R in fourth place.

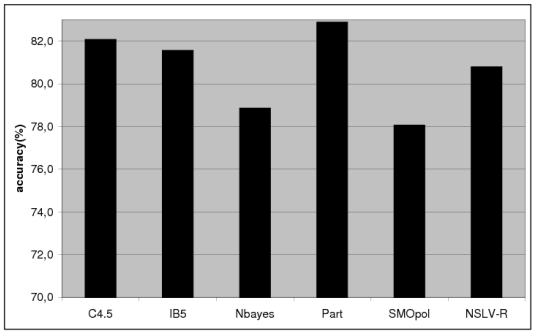

Fig. 8. Comparison of the expected accuracy of classical classifiers.

5.2. *Comparing NSLV-R with other fuzzy-rule-learning algorithms in terms of accuracy.*

In this section we compare NSLV-R with other well-known fuzzy-rule-learning algorithms using the KEEL platform [27] and the settings recommended by KEEL for all algorithms. The fuzzy algorithms used are as follows [*]:

- GFS-GCCL-C (*Fuzzy rule approach based on a genetic cooperative-competitive learning*) [28]: This genetic algorithm handles each fuzzy rule as an individual and assigns a fitness value to each rule. It uses linguistic values with fixed membership functions as antecedent fuzzy sets, thus meaning that a linguistic interpretation of each fuzzy rule is easily obtained.

[*]We use in this experimentation the "short name" that KEEL platform associates to these algorithms.

Table 9. Test accuracy for NSLV-R and several well-know fuzzy rule based learning algorithms

Databases	GFS-GP-C	GFS-GPG-C	GFS-GCCL-C	GFS-LogitBoost-C	GFS-SP-C	NSLV-R
ann	82.4 (3)	77.9 (5)	87.1 (2)	76.2 (6)	80.2 (4)	95.1 (1)
aut	51.7 (3)	41.5 (6)	57.1 (2)	47.3 (4)	46.8 (5)	70.7 (1)
bal	78 (4)	70.1 (5)	84.1 (2)	87.2 (1)	69.4 (6)	80.2 (3)
bpa	55.3 (6)	60.8 (4)	58.3 (5)	70.2 (2)	63.5 (3)	71 (1)
brd	55.1 (4)	56.9 (3)	51.7 (6)	57.8 (2)	54 (5)	65 (1)
car	78.7 (3)	76.7 (5)	70 (6)	86.7 (2)	77.4 (4)	87.5 (1)
col	85.9 (1)	83.7 (4)	71.8 (5)	70.6 (6)	84 (3)	84.2 (2)
crx	84.9 (2)	84.5 (3)	73 (6)	82.9 (5)	85.1 (1)	83.9 (4)
drm	76.7 (3)	68.3 (4)	81.4 (2)	30.6 (6)	58.2 (5)	95.6 (1)
gls	45 (6)	54.4 (4)	59.9 (2)	67 (1)	48.5 (5)	57.3 (3)
h-c	78.2 (1)	73.5 (6)	77.9 (2)	76.2 (4)	76.88 (3)	75.2 (5)
h-h	66.3 (1)	62.9 (6)	63.9 (5)	64.3 (4)	64.9 (3)	66.2 (2)
h-s	73 (5)	78.9 (1)	78.1 (2)	74.8 (3)	74.1 (4)	72.9 (6)
hpt	81.9 (3)	82 (2)	80.6 (4.5)	78.7 (6)	80.6 (4.5)	85.7 (1)
irs	86.7 (6)	90.7 (5)	95.3 (2)	95.3 (2)	95.3 (2)	93.9 (4)
pim	73.6 (4)	73.8 (2.5)	68.8 (6)	75.9 (1)	72.5 (5)	73.8 (2.5)
son	70.7 (2)	61.1 (5)	61.6 (4)	51.9 (6)	69.1 (3)	72.9 (1)
tao	82.8 (3)	76.2 (6)	81.2 (5)	87.0 (1)	86.9 (2)	82.3 (4)
thy	85.5 (6)	88.8 (3)	87.0 (5)	96.3 (1)	88.5 (4)	92.6 (2)
trf	76.5 (2.5)	76.2 (5.5)	76.2 (5.5)	78.9 (1)	76.5 (2.5)	76.4 (4)
vot	94.7 (2)	93.6 (3.5)	61.4 (6)	87.1 (5)	93.6 (3.5)	96.5 (1)
wdbc	93.2 (1)	81.9 (4)	92.6 (2)	72.8 (5)	71.2 (6)	92.2 (3)
wpbc	76.8 (4)	88.2 (2)	76.4 (5)	96.6 (1)	80.9 (3)	71.2 (6)
zoo	84.2 (3)	82.5 (4)	91.6 (2)	41.9 (6)	79 (5)	96.4 (1)
Average	75.8	74.4	74.5	73.1	74.0	80.8

- GFS-GPG-C (*Fuzzy Learning based on Genetic Programming Grammar Operators*) [29]: This classifies an instance as a result of a quadratic combination of its features. The weights of this combination are fitted as a quadratic discriminant. An instance is classified with the class that has the better value for the quadratic combination of its features.
- GFS-SP-C (*Fuzzy Learning based on Genetic Programming Grammar Operators and Simulated Annealing*) [29]: A Simulated Annealing algorithm is used to learn a fuzzy classifier. The number of labels and number of rules must be given for each hypothesis. Likewise, as it is possible to manage any combination of conjunction and/or disjunctions in the antecedent part of a fuzzy rule, a maximum deep tree size must be given too. These parameters, in conjunction with Simulated Annealing type parameters, are of vital importance in the evolution of this method.
- GFS-LogitBoost-C [30]: LogitBoost, or the logistic extended additive model, is a back-fitting algorithm which repeatedly invokes a learning algorithm to successively generate a committee of simple, low-quality classifiers. In this algorithm, each of the weak hypotheses is a Fuzzy rule extracted from data. These fuzzy rules are extracted from data by means of a genetic algorithm. Each time a new simple classifier is added to the compound one, the examples in the training set are reweighted (so that future classifiers will focus on the most difficult examples).
- GFS-GP-C (*Fuzzy Learning based on Genetic Programming*) [29]: This combines genetic programming operators with a Simulated Annealing Search to evolve fuzzy-rule-based classifiers.

Table 9 shows a comparison of the above-mentioned fuzzy methods.

In a similar way to the previous subsection, we apply the Friedman's and Holm's tests for finding significant differences among the proposed methods. In Table 10 is showed the Friedman's test. It shows that there are significant differences between the methods since the statistic is greather than the critical value.

Table 10. Friedman statistic (distributed according to chi-square with 5 degrees of freedom)

Statistic	critical value	p value
11.40472	11.0705	0.04392

The average ranking of the studied algorithms are showed in Table 11.

Table 11. Average rankings of the algorithms (Friedman)

Algorithm	Ranking
NSLV-R	2.5208
GFS-GP-C	3.2708
GFS-SP-C	3.8125
GFS-LogitBoost-C	3.375
GFS-GCCL-C	3.9167
GFS-GPG-C	4.1042

Table 12. Holm's test comparison Table for $\alpha = 0.05$

i	algorithm	z	p	α/i
5	GFS-GPG-C	2.931	0.00337	0.01
4	GFS-GCCL-C	2.584	0.00975	0.0125
3	GFS-SP-C	2.391	0.01677	0.016667
2	GFS-LogitBoost-C	1.581	0.113739	0.025
1	GFS-GP-C	1.388	0.164915	0.05

We now apply Holm's test to compare the best ranking method (NSLV-R) with the remaining methods. Table 12 presents these results. In this table, the methods are ordered with respect to the z value obtained. Holm's test rejects the hypothesis of equality with the rest of the methods ($p < \alpha/i$). Analyzing the statistical study shown in Table 12 we conclude that NSLV-R is significatively better than *GFS-GPG-C* and *GFS-GCCL-C* and there are no significant differences with respect to *GFS-GP-C*, *GFS-SP-C* and *GFS-LogitBoost-C*. However, if we compare these algorithms taking into account the average expected accuracy (see Figure 9), we can see that NSLV-R gives by far the best value, with a difference to the second best algorithm of a 5%.

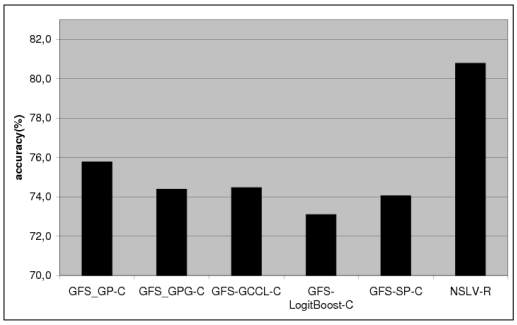

Fig. 9. Comparison between expected accuracy of fuzzy classifiers.

6. Conclusions

We have presented a learning algorithm that can be used to obtain fuzzy relational rules. The basic idea is to provide the learning algorithm with a more descriptive capacity in the rule model. However, this increased descriptive capacity means more complexity, therefore we have defined a prior procedure to determine the most relevant relations to be considered by the learning algorithm. This procedure involves the use of two filters: a basic filter, which allows the use of expert information and/or particular information concerning the universe of the variables to reduce the number of relations that the learning algorithm can consider, and a heuristic filter based on the use of a measure of information that allows us to define the most relevant relations given the classification variable of the problem. Finally, the experimental study shows that the inclusion of fuzzy relational rules is interesting and provides a clear improvement in accuracy with respect to other fuzzy-rule-learning algorithms.

This work is, however, simply a first step, and it will therefore be necessary to experiment with more relations and to improve the efficiency of the process as a whole. With this in mind our next goals are, first getting to applied the algorithm to specific problems to see the contribution that the FRR have on these problems, and on the other hand, include more relations in the learning algorithm and ensure that this does not produce a worsening of the performance of the algorithm.

Acknowledgment

This work is partially supported by the projects P09-TIC-04813 and I+D+i TIN2007-66367.

References

1. E.H. Mamdani, S. Assilian , "An experiment in linguistic synthesis with fuzzy logic controller," International Journal of Man-Machine Studies, **7**, 1-13 (1975).
2. T. Takagi, M. Sugeno, "Fuzzy identification of systems and its application to modeling and control," IEEE Transactions on Systems, Man and Cybernetics, **15**, 116-132 (1985).
3. E. Aguirre, A. González, Raúl Pérez, "An inductive approach for learning fuzzy relational rules," Proceedings Third EUSFLAT 2003 Conference, 88-93 (2003).
4. A. E. Gaweda, J.M. Zurada, "Data-driven linguistic modeling using relational fuzzy rules," IEEE Transactions of Fuzzy Systems, vol **11**, no. 1, 121-134 (2003).
5. R.R. Yager, "The representation of fuzzy relational production rules," Journal of Applied Intelligence, **1**, 35-42 (1991).
6. A. González, R. Pérez ,"Completeness and consistency conditions for learning fuzzy rules,". Fuzzy Sets and Systems, **96**, 37-51 (1998).
7. Y. Caises, E. Leyva, A. González, R. Pérez, "A Genetic Learning of Fuzzy Relational Rules," Proceedings of the IWCCI 2010, EEE World Conference on Computational Intelligence- FUZZ-IEEE, 207-214 (2010).
8. A. González, R. Pérez,"Improving the genetic algorithm of SLAVE," Mathware and Soft Computing, **16**, 59-70 (2009).
9. R.R. Yager, D. P. Filev, "Relational partitioning of fuzzy rules," Fuzzy Sets and Systems, **80**, 57-69 (1996).
10. T. Mitchell, "Machine Learning," Ed. MacGraw-Hill (1997).
11. Michalski R.S., "Understanding the nature of learning," Machine Learning: An artificial intelligence approach (Vol **II**). San Mateo, CA: Morgan Kaufmann, 1986.
12. A. González, R. Pérez, "An analysis of the scalability of an embedded feature selection model for classification problems," Proceedings of the Information Processing and Management of Uncertainty on Knowledge-Based Systems IPMU 2006, 1949-1956 (2006).
13. A. González, R. Pérez, "Selection of Relevant Features in a Fuzzy Genetic Learning Algorithm". IEEE Transactions on System, Man, and Cybernetics Part B, vol. **31** (3), 417-425 (2001).
14. S. Kullback, "Information Theory and Statistics," Gloucester, Mass, 1978.
15. L. Wehenkel, "On uncertainty measures used for decision tree induction," Proc. of IPMU'96, 413-418 (1996).
16. A. Asuncion, D.J. Newman, "UCI Machine Learning Repository" [http://www.ics.uci.edu/ mlearn/MLRepository.html]. Irvine, CA: University of California, School of Information and Computer Science, 2007.
17. A. González, R. Pérez, "SLAVE: to genetic learning system based on an iterative approach," IEEE Transaction on Fuzzy System, vol. 27 (2), pp. 176-19, 1999.
18. O. Dunn,"Multiple Comparisons among means," Journal of the American Statistical Association, 1961.
19. I.H. Witten and E. Frank, "Data mining: practical machine learning tools and techniques, " Morgan Kaufmann, San Francisco, 2nd edition (2005).
20. J. Demsar,"Statistical comparisons of classifiers over multiple data sets," Journal of Machine Learning Research, **7**, 1-30 (2006).
21. M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, Journal of the American Statistical Association, vol. 32, no. 200, (1937).
22. Quinlan J.R., "C4.5 Program for Machine Learning," Morgan Kaufmann (1993).
23. D.W. Aha, D.F Kibler and M.K. Albert, "Instance-based learning algorithms. Machine Learning", **6**, 37-66 (1991).
24. G.H. John and P. Langley, "Estimating continuous distributions in bayesian classifiers," 11th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, 338-345 (1995).
25. E. Frank and I.H. Witten, "Generating accurate rule sets without global optimization," Proceedings of the 15th International Conference on Machine Learning, Morgan Kaufmann, 144-151 (1998).
26. J. Platt, "Fast training of support vector machines using sequential minimal optimization," Advances in Kernel Methods - Support Vector Machine. MIT Press

(1998).
27. J. Alcalá et al, KEEL: "A Software Tool to Assess Evolutionary Algorithms to Data Mining Problems," Soft Computing **13**, 307-318 (2009).
28. H. Ishibuchi et al, "Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems," IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics **29** (5), 601-618 (1999).
29. L. Sánchez, I. Couso, I., "Combining GP Operators With SA Search To Evolve Fuzzy Rule Based Classifiers," Information Sciences **136** (1-4), 175-192 (2001).
30. J. Otero, L. Sánchez, "Induction of descriptive fuzzy classifiers with the Logitboost algorithm," Soft Computing **10** (9), 825-835 (2006).