ELSEVIER International Journal of Approximate Reasoning 19 (1998) 193–220

INTERNATIONAL JOURNAL OF APPROXIMATE REASONING

A fuzzy theory refinement algorithm [1]

Antonio Gonzalez *, Raúl Perez

Departamento de Ciencias de la Computación e Inteligencia Artificial ETS de Ingeniería Informática, Universidad de Granada, 18071 Granada, Spain

Received 1 April 1997; accepted 1 October 1997

Abstract

A fuzzy theory refinement algorithm composed of a heuristic process of generalization, specification, addition and elimination of rules is proposed. This refinement algorithm can be applied to knowledge bases obtained from several sources (learning algorithms, experts), but its development is strongly associated with the SLAVE learning system. SLAVE was developed for working with noise-affected systems where the application of some conditions of classical learning theory do not produce good descriptions. This learning system allows us to obtain the structure of the rule, i.e. it can determine among all the variables proposed those that are relevant for describing the system (feature selection). SLAVE uses an iterative approach for learning with genetic algorithms. This method is an alternative approach to the classical Pittsburgh and Michigan approach and it consists in obtaining a useful rule to describe the system in each iteration. So in this approach, the final solution is obtained from partial solutions. The refinement module appended to SLAVE (SLAVE + R) is proposed as a method for verifying that the union of the partial solutions is a good global solution. Furthermore, this module allows us to minimize the number of necessary rules, maintaining or improving the accuracy and understanding of these rules.

Keywords: Machine learning; Fuzzy logic; Fuzzy rules refinement

* Corresponding author. Tel.: +34 58 243 199; fax: +34 58 243 317; e-mail: a.gonzalez@decsai.ugr.es.

[1] This work has been supported by CICYT under Project TIC95-0453.

PII: S0888-613X(98)00013-9

1. Introduction

Knowledge acquisition is one of the main problems in developing knowledge-based systems. Two phases in the acquisition knowledge problem may be distinguished: in the first stage, the knowledge is directly given by an expert or it is automatically extracted using a learning algorithm [7,18,20]; in the second stage, this knowledge must be refined in order to produce a high-performance system (theory refinement). In a similar way, this process for refining the initial knowledge can be automatically done by taking into account a set of examples that describes the behavior of the system in the past. There are several ways to represent the obtained knowledge but this knowledge is usually expressed using rules that represent the relationships between the different variables in the problem. Therefore, given an incomplete and/or incorrect fuzzy rule base (fuzzy theory) that represents the initial knowledge (where this set can be empty) and a set of consistent examples, the refinement problem consists in modifying this initial set of rules so that it may be better adapted to the example set.

The basic operations in this type of algorithm, when the knowledge is represented by rules, are the processes of specification, generalization, addition and elimination of rules. Consequently, the main difference between two algorithms of rule refinement is the heuristic used by the algorithm for applying these processes. For example, in the EITHER algorithm [19], the set of rules is improved, causing minimum changes to the initial knowledge. However, we can find the JoJo algorithm [4] where its heuristic is only guided by the accuracy of the rule set over the example set. Both systems were designed for working with crisp rules.

In [7] we presented a learning algorithm of fuzzy rules called SLAVE. This algorithm obtains a rule set for describing the consequent variable, using an iterative method. The selection of the best rule in each iteration is done by a genetic algorithm (GA). An inconvenience of the iterative method is that the solution of the learning problem is composed of a set of "good" individual rules, but the behavior is unknown when all the rules work together. Furthermore, SLAVE uses a graded definition on the consistency condition with two parameters to determine the good quality of the rule. We have proven experimentally that there is a strong dependency between the learning rates and the values of these parameters. This strong dependency implies that a successful estimation of these parameters is needed to obtain good results with SLAVE, but the estimation of the consistency parameters are not easy.

For these reasons, we have developed a refinement algorithm associated to SLAVE as a new component. So, this algorithm was proposed as a method of verifying that the union of the individual rules is a good global solution and of reducing the dependency with the learning parameters. In other studies such as [10–12], an initial version of a refinement algorithm for crisp and fuzzy

consequent domains was presented, and in [14] we proved that this refinement algorithm can be successfully applied to the rule set obtained with other learning algorithms.

The basic elements of this algorithm are as follows.

(a) *The rule model:* This element is inherited from SLAVE and it is an important feature of this system because it allows us to determine the relevant variables for each rule. Therefore it is strongly associated with determining the structure of the rule and obtaining more understandable descriptions.

(b) *The heuristic:* The special way in which the process of generalization, specification, addition and elimination of rules are combined.

(c) *The inference process:* We can distinguish two different kinds of inference processes, one for crisp consequent domain and another for fuzzy consequent domain. Both types of learning problem can be solved using SLAVE. For the first one, SLAVE uses a particular function to establish the compatibility between an example and the rule set and a special procedure for solving the conflict problems. The mechanism for solving conflicts uses a rule sorting criterion based on a measure of the accuracy of each rule. Therefore, the refinement algorithm supposes that the rule set given as input is ordered using this criterion. For problems with fuzzy consequent domain, the order of the rule is not important since the output is obtained by the combination of the activated rules. In this case, a simple inference process and a defuzzification method are used.

In this paper, we propose an improved version of this refinement algorithm, where the main differences when it is compared to the initial version are:

1. a modification of the heuristic that guides the algorithm,
2. a new process that modifies the semantics of the labels of the consequent variable and it allows the introduction of a better system description in the fuzzy consequent domain.

The aim of Section 2 is to briefly show a description of the SLAVE learning algorithm and to consider the initial ideas for the refinement algorithm. Section 3 describes the refinement algorithm for the crisp consequent domain and its behavior is tested using other learning algorithms. In a similar way, Section 4 describes and tests the refinement algorithm for the fuzzy consequent domain. Finally, we show the conclusions of this work in Section 5.

2. An overview of the slave learning algorithm

SLAVE is an inductive learning algorithm that was initially proposed in [7] and later developed in [8,9], and uses fuzzy rules to represent the knowledge obtained. The basic element of the SLAVE learning algorithm is its rule model

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

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 general, we can consider the variables in this model to be linguistic, 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 elements 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 is clarified in the following example:

Example 2.1. Let X_i be a variable whose domain is shown in Fig. 1(a). An antecedent like

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

is equivalent to

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

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

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

Another important characteristic of this learning algorithm is that it uses the iterative genetic approach in order to learn the rule set. The iterative approach is presented as an alternative to Michigan and Pittsburgh's well-known approaches for machine learning. This approach consists in including a genetic algorithm in an iterative scheme similar to the following [13]:

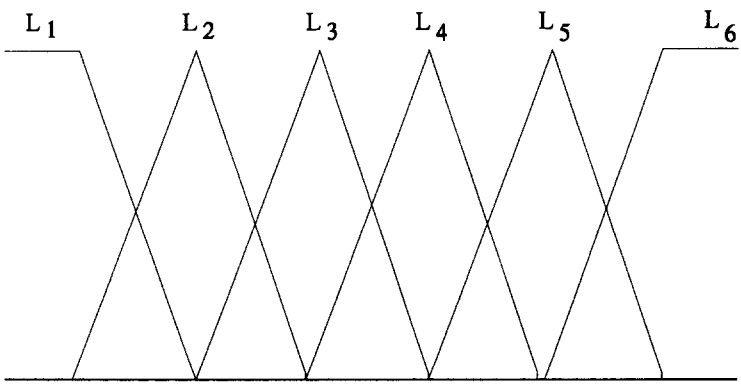

Fig. 1. A fuzzy domain.

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 sufficient to represent the examples in the training set, the system will return the set of rules as the solution. Otherwise return to step 1.

The GA obtains a rule in each step that is a partial solution to the learning problem. The true solution is obtained by appending each rule to the rule set. In recent literature, we can find different algorithms that use this new approach, such as [3,7,23]. An interesting discussion about the problems generated by the use of these approaches can be seen in [13].

Therefore, in the iterative approach, the GA is used for selecting the best rule in each iteration of the learning process. This GA and its parameters are described in [9]. In SLAVE, in order to measure the good quality of the rule, we introduced two definitions that are based on the classical consistency and completeness conditions. These conditions provide the logical foundation of the algorithms for concept learning from examples.

Definition 2.1. The completeness condition states that every example in some class must verify some rule from this class.

Definition 2.2. 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 entire rule set, but 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 propose a completeness degree and a consistency degree; both definitions use the concepts proposed in [9].

Definition 2.3. The completeness degree of a rule $R_B(A)$ is defined as

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

where n_B is the number of examples of the B class and $n^+(R_B(A))$ is the number of positive examples covered by $R_B(A)$.

The soft consistency degree is based on the possibility of admitting some noise in the rules. Therefore, in order to define the soft consistency degree we use the following set:

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

that represents the set of rules having a number of negative examples ($n^-(R_B(A))$) strictly less than a percentage (depending on k) of the positive examples ($n^+(R_B(A))$).

Definition 2.4. The degree to which a rule R satisfies the soft consistency condition is

$$\Gamma_{k_1k_2}(R) = \begin{cases} 1 & \text{if } R \in \Delta^{k_1} \\ \dfrac{k_2 n^+(R) - n^-(R)}{n^+(R)(k_2 - k_1)} & \text{if } R \notin \Delta^{k_1} \text{ and } R \in \Delta^{k_2}, \\ 0 & \text{otherwise,} \end{cases}$$

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

This definition uses two parameters: k_1 is a lower bound of the noisy threshold and k_2 is an upper bound of the noisy threshold. In both definitions, we have used the definitions of positive and negative example number proposed in [9]. These definitions are based on the use of the cardinality of two fuzzy sets (the positive and negative example sets).

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

$$\max_{A \in D}\{\ A(R_B(A)) \ \times \ \Gamma_{k_1k_2}(R_B(A))\},$$

where $D = P(D_1) \times P(D_2) \times \cdots \times P(D_n)$. Fig. 2 shows the different elements of the SLAVE learning algorithm and the relationships between them.

In our practical experience, we have detected some problems in the good quality of the rule set obtained by SLAVE. The causes of these problems are mainly due to the following reasons:

(a) The iterative approach of SLAVE obtains each rule from the rule set individually, therefore the final rule set contains a set of good individual rules that have an unknown behavior when they are all used to classify an example.

(b) The rule set obtained by SLAVE has a strong dependency on the parameters of the consistency conditions. Therefore, a good estimation of these parameters is necessary for correct learning.

So, the refinement algorithm is proposed as a method to verify that the union of the individual rules is a good global solution and to reduce the dependency on the consistency parameters. Furthermore, this module allows us to minimize the number of necessary rules, maintaining accuracy, and to improve the understanding of these rules. SLAVE+R is the name which we give to the system composed of SLAVE with the addition of the refinement module, and its basic structure is showed in the Fig. 3.

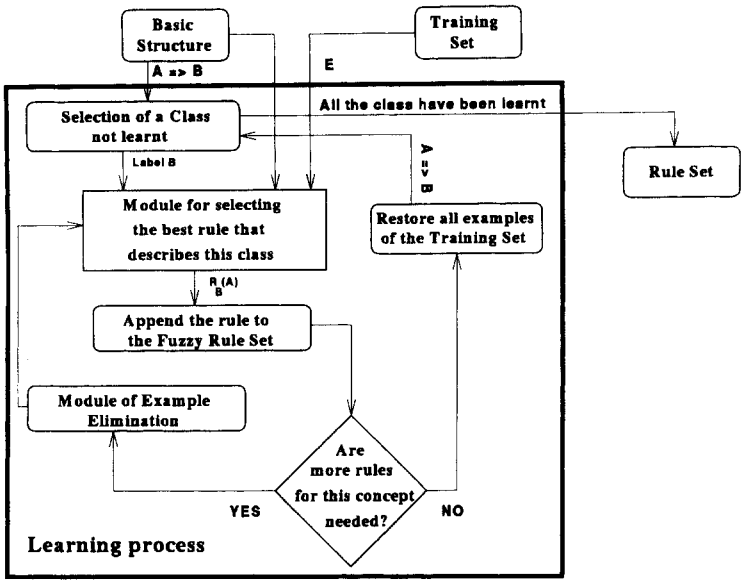

Fig. 2. Description of Learning Process.

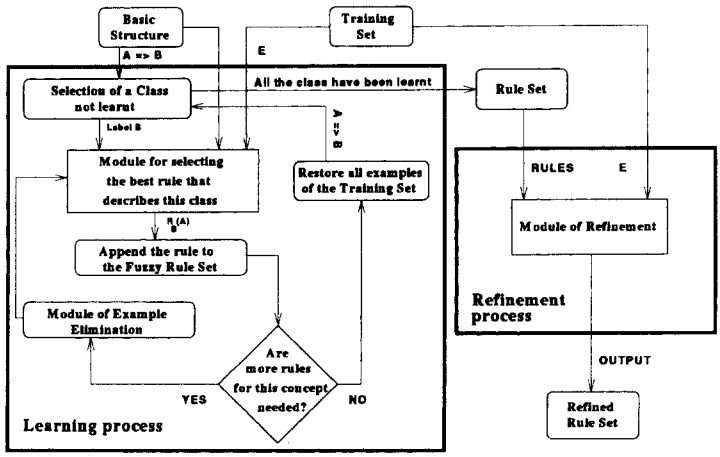

Fig. 3. Description of SLAVE + R.

The refinement algorithm is a heuristic algorithm that tries to maximize the good quality of the rules. The estimated good quality of the rule set is based on error measurements. The algorithm uses the hill-climbing strategy and it is composed of the operations of specification, generalization, addition and elimination of rules that are repeated until the global error and the number of rules cannot be decreased.

The refinement uses a heuristic function to select the most promising action in each step of the algorithm in order to find a satisfactory solution. This heuristic function measures the error produced by a set of rules and therefore the heuristic function is closely related to the inference process. Because the inference process for the crisp consequent is different to the fuzzy consequent, we develop two different refinement modules that use similar steps but with a different interpretation. In the following sections, we describe each one of these modules.

3. Refinement module for crisp consequent domain

The refinement module attempts to improve the rule set obtained by SLAVE, reducing the problems previously described. The module is composed of two tasks. The first one consists in improving the correctness of each rule. For this purpose, a specification process is used which attempts to ensure that each rule covers the highest number of well-classified examples from the original rule without covering its worst-classified examples. Following this task, it is possible that some of the worst-classified examples covered by some rules become unclassified examples. The next task attempts to cover these unclassified examples using a generalization process on the existing rules or additional new rules. The previous tasks are repeated on the rule set until a termination condition is verified. Fig. 4 shows the different steps of this algorithm.

The refinement uses a heuristic function and a hill-climbing strategy to select the most promising action in each step of the algorithm in order to find a suitable solution. We consider a function that measures the global precision of the current rule set on the training set. Thus, in order to define this function it is necessary to describe the predictive module used. The inference process begins with an ordered rule set and the classification of an example is done in the following way: the adaptation between the example and the antecedent part of each rule is evaluated and the rule class with the best adaptation is returned. If there is more than one rule with the best adaptation (conflict problem), the class from the rule with the lowest order in the rule set is returned. The heuristic component of the refinement algorithm selects rules in the order previously described in the rule set. However, we have not considered any special ordering for variables and values. They are taken by considering the default order.

Now, we will describe each one of the steps in the refinement algorithm with more detail.

(1) *Improving the precision of the rules:* In this step, we try to improve the prediction capacity of the rule set obtained by SLAVE on the training set, increasing the correctness for each rule.

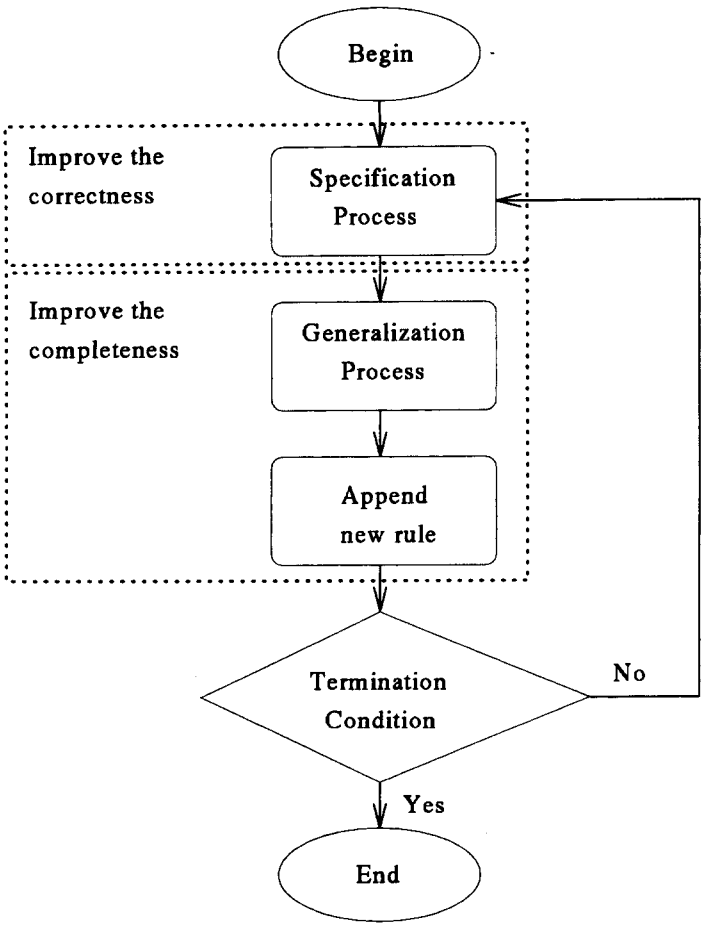

Fig. 4. Refinement for crisp consequent domain.

Definition 3.1. A rule is correct if its precision, using the predictive process, is complete, i.e. this rule has total success.

To obtain a rule set where all the rules are correct is not appropriate in some cases, for example, when we work with noisy training sets since a subset of the examples will be correctly classified but we cannot say anything about the rest. A possible solution consists in grading the correctness of the rule. This degree is based on the precision of the rule, i.e. the relation between successes and successes plus failures produced by the rule on the training set. Using this grading in the completeness concept, the goal of the refinement algorithm consists in finding a good rule set from the original rule set, in which each rule must be as complete as possible.

When we learn with SLAVE, the parameters of the consistency degree fix an estimation on the minimum degree of correctness of each rule that it is learnt. One of the problems that we must solve when we work with SLAVE, consists in assigning suitable values for the k_1 and k_2 parameters. If the choice of these values is not correct, during the rule selection, rules with a lower degree of correctness may be preferred over other rules with a high correctness degree but lower valuation. The following example shows this problem.

Example 3.1. Let us suppose that we want to learn the example set shown in Fig. 5 using SLAVE. In this problem, there are two predictive variables X_1 and X_2, for which the domains are $\{a_1, a_2, a_3, a_4\}$ and $\{b_1, b_2, b_3, b_4\}$ respectively. The examples can be members of two different classes $\{+, -\}$. Furthermore, let us suppose that $k_1 = k_2$ and that they are fixed at $\frac{1}{3}$. Under these conditions, the first rule that is returned by SLAVE for (+) class is

$$\text{IF } X_1 \text{ is } \{a_1, a_2, a_3\} \quad \text{and} \quad X_2 \text{ is } \{b_1, b_2, b_3\} \text{ THEN } +$$

for which its value is

$$A(R) = \frac{7}{8},$$

$$\Gamma_{k_1 k_2}(R) = 1.0,$$

$$A(R)\ \Gamma_{k_1 k_2}(R) = \frac{7}{8}$$

and its precision is $\frac{7}{9}$. Contrary to this rule, we can find the rule

$$\text{IF } X_1 \text{ is } \{a_1, a_2, a_3\} \quad \text{and } X_2 \text{ is } \{b_1, b_2\} \text{ THEN } +$$

for which its value

$$A(R) = \frac{6}{8},$$

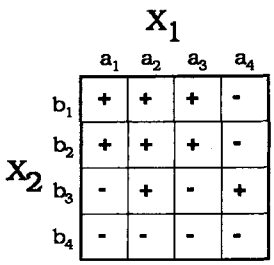

Fig. 5. A set of examples.

$$\Gamma_{k_1k_2}(R) = 1.0,$$
$$A(R)\ \Gamma_{k_1k_2}(R) = \frac{6}{8}$$

is less than in the previous rule, but the rule is correct.

SLAVE has a tendency to generalize the rules as far as the k_1 and k_2 parameters allow. The first step of the algorithm attempts to counteract this behavior by specifying the rules for reducing the number of failures in each rule. Thus, this process begins by specifying the rules in the order in which they appear in the rule set. We shall say that a value for a variable is active if this value appears for that variable in the description of the rule. So, we shall say that a value for a variable is not active if this value does not appear for that variable in the description of the rule. In this way, a specification operation can be considered as changing to non-active value that previously was active and we will say that this value is deactivated. This process can be described in the following way.

1. The most relevant rule is selected.
2. Select a variable and do the following:
   2.1. Select an active value.
   2.2. This value is deactivated.
   2.3. If the overall accuracy is improved or maintained, the modification is accepted. Otherwise, the value is activated.
   2.4. If there are other active values, go to 2.1.
   2.5. If all the values of the variable are deactivated, then this rule is removed from the rule set.
   2.6. If there are other unselected variables, go to 2.
3. If there are more unselected rules, select the next most relevant rule and go to 2, otherwise the process finishes.

Furthermore, we must note that this step allows us remove rules from the rule set. Therefore, the elimination of a rule from the rule set is considered as a particular case of specification and it is caused when all the values of an antecedent variable are deactivated.

(2) *Improving the completeness generalizing the rules:* In this second step, the refinement algorithm attempts to improve the completeness of the rules by using an ordered process of generalization. In this process, new values are appended to the antecedent variables for each rule. This generalization is done when the overall accuracy is kept or strictly improved, using the rule set that includes the modified rule. There are two reasons for the improvement in accuracy:

(a) Unclassified examples can be correctly covered by generalizing one of the rules.

(b) Examples incorrectly classified by one rule with a lower order can be correctly covered when a rule with a higher order is generalized.
Consequently, this step of the algorithm improves the completeness and the correctness of some of the rules of the rule set.

This procedure has a similar structure to the previous algorithm, but the order of rules is reversed and it begins by selecting the least relevant rule:

1. The least relevant rule is selected.
2. Select a variable and do the following:
   2.1. Select a deactivated value.
   2.2. This value is activated.
   2.3. If the overall accuracy is strictly improved or maintained, the modification is accepted. Otherwise, the value is deactivated.
   2.4. If there are other deactivated values, go to 2.1.
   2.5. If there are other unselected variables, go to 2.
3. If there are more unselected rules, select the next most relevant rule and go to 2, otherwise the process finishes.

In this process, the inverse order is used for the following reason: the inference method used favors the rules that appear in the first positions in the rule set when the conflict problems appear. A rule that has the T position in the rule set may include in its description the descriptions of all the rules that have a lower position than T, if it does not reduce the accuracy of the systems. This fact means that the first rules are more specific than the last rules in the rule set. So, the last rules have a higher probability of being generalized than the first rules.

(3) *Improving the completeness by the addition of new rules:* The next step in the algorithm, consists in appending new rules to cover the examples that are not covered by any rule and which the previous process cannot cover. This task adds the most specific rule with the best adaptation for each example that is not covered in the training set.

Example 3.2. Let us suppose we have three variables, X_1, X_2 and X_3, and the fuzzy domain associated with each one is shown in Fig. 6, and furthermore, let us suppose that the example (r_1, r_2, r_3) is not covered by any rule, this example being a member of a certain B class. In this case, the most specific rule with the best adaptation with this example is

$$\text{IF } X_1 \text{ is } \{A_{13}\} \text{ and } X_2 \text{ is } \{A_{23}\} \text{ and } X_3 \text{ is } \{A_{31}\} \text{ THEN Y is } B$$

(4) *Termination condition:* The previous steps are repeated until the rule set is stable. In the algorithm, the rule set is determined as being stable when the number of rules and the accuracy is maintained in two consecutive iterations.

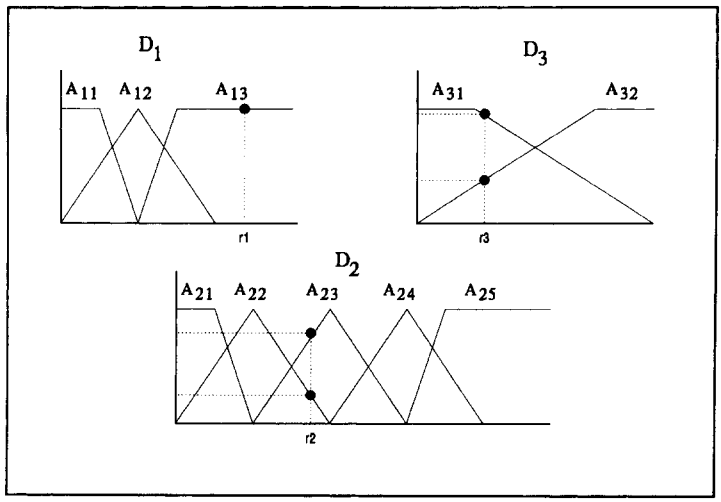

Fig. 6. Domains of the X_1, X_2, X_3 variables.

3.1. Testing the refinement algorithm in SLAVE

The main problem that we encounter when we want to use SLAVE is selecting an appropriate value for the parameters of the consistency degree. In this paper, we try to prove that SLAVE with this refinement algorithm greatly reduces the dependency on the parameters of the weak consistency condition.

In order to test the performance of the refinement algorithm, we use a database of examples that we split into a training set (70%) and a test set (30%). We take 200 different values for the parameter of the consistency condition in the following four situations:

(A) SLAVE using the $[k, k]$-consistency degree.

(B) SLAVE using the $[0, k]$-consistency degree.

(C) SLAVE using the $[k, k]$-consistency degree and the refinement algorithm.

(D) SLAVE using the $[0, k]$-consistency degree and the refinement algorithm.

The values of k are taken every 0.005, beginning at 0 and finishing at 1 (200 points). For each situation, we draw a line connecting the different points obtained.

For this experiment, we have selected the well-known IRIS plants database, described by Fisher in [6]. The problem of IRIS plants lies in classifying the plants in function of four predictive variables. The predictive variables are continuous and the classification variable is discrete with three different classes. In the overall databases, there are 150 examples, 50 for each class.

When working with SLAVE, we need to define the domain on the continuous variables using fuzzy labels. In this problem, we have used five fuzzy labels for each antecedent variable, distributed uniformly over each range.

In Fig. 7 the different graphs obtained for situations A, B, C and D are shown, respectively. In Table 1, we show the probability of obtaining the best result in the training and test sets (P. Train. and P. Test) and the best results in each case (Best Train. and Best Test). From the table and the graphs, we can obtain the following conclusions:

- The $[0,k]$ model for the parameter of the consistency degree presents a higher probability of finding the best values in test and training than the model $[k,k]$. In this way, the first of them has less dependence over the estimation of its parameter. Furthermore, for this model the range of variation from the results is shorter (both in training and test).
- Using the refinement algorithm, the results are improved in both cases, reducing the range of variation.
- The best result is obtained when the model with two parameters and the refinement algorithm are used. In these cases, the best results in the test set are obtained (95.04%) and those with high probability may be found (74%).

We can then consider that the refinement algorithm using the model $[0,k]$ produces the most promising results, giving more stability to the behavior of the rule set and reducing the dependency on the estimation of the parameter. From now on, in view of the previous result, we use the [0,1] model and the refinement algorithm that is proved to obtain high accuracy in the majority of the situations.

In the next section, we compare the result obtained by SLAVE with the refinement algorithm and the [0, 1] model with other classical methods of learning.

3.2. Comparing SLAVE+R with other learning algorithms

In this section, we compare the results obtained by SLAVE+R with other learning algorithms on different databases. The learning algorithms considered are C4.5, C4.5rules [21] and CART [2]. C4.5 and CART return decision trees for classifying the examples and C4.5rules generates a set of rules from the decision trees returned by C4.5.

The databases used in the experimental work are the following:

- The MONK's problems [21,22] are a collection of three classification problems on a discrete domain with six attributes.
- The IRIS database [6] used in the previous section. We have considered five uniformly distributed linguistic labels for each antecedent variable.

In the IRIS database, we have taken five different partitions (training (70%) and test (30%)) for each one. In Table 2, we show the average accuracy on the test set for each database. On the MONK's problems, we take the same training and test sets proposed by Quinlan [21]. From Table 2 we can deduce:

- For the MONK's problems, where all the predictive variables are nominals, SLAVE + R obtains the best results. Only on MONK1, does C4.5rules obtain as good a result as SLAVE + R.

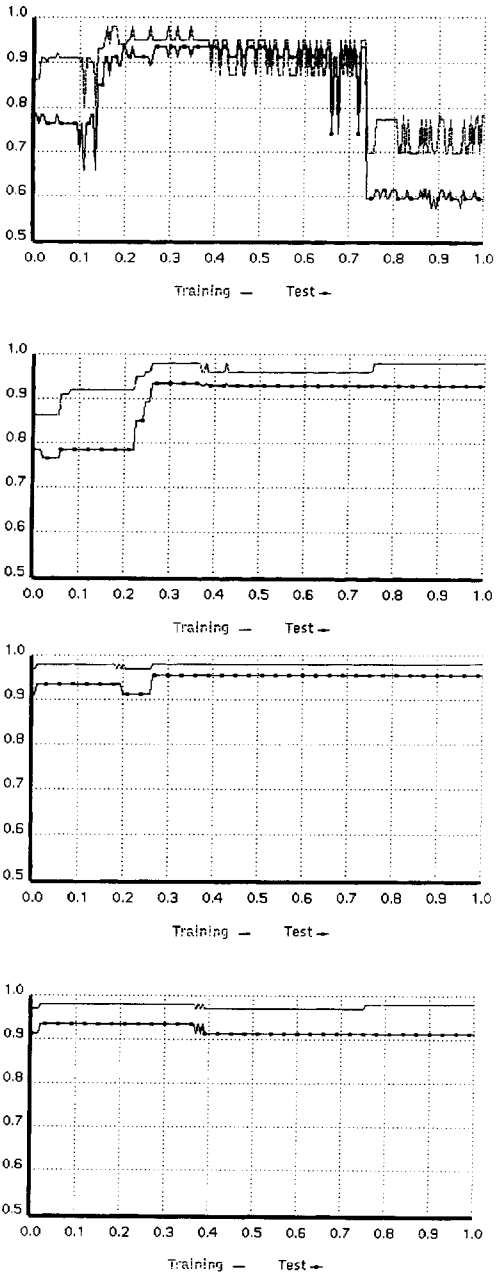

Fig. 7. Graphs of the results of the experiments.

Table 1
Probability of finding the best result

Exp.	P. Train. (%)	P. Test (%)	Best Train. (%)	Best Test (%)
A	4	31.5	98.06	93.20
B	39	70.5	98.06	93.20
C	65	38.5	98.06	93.20
D	93	74	98.06	95.04

- For the IRIS database, where all the predictive variables are continuous, the best result is returned by CART. SLAVE + R is the second best result which is very close to the CART result. SLAVE + R presents better results than the classical learning algorithms C4.5 and C4.5rules.

4. Refinement module for fuzzy consequent domain

When the domain of the consequent variable is continuous, we can discretize the domain using a fuzzy domain. In this case, we have a fuzzy consequent domain. Contrary to the refinement module for the crisp consequent, this refinement algorithm has two components that are specially designed for these types of problems. These special components are the inference process and error measurement. The predictive process used is the max–min inference system and the average of the center of gravity weighted by the matching value as a defuzzification method and the minimum operator as t-norm [5]. Based on the inference system, we define a measure for establishing the accuracy of the rule set, using an error measurement. This error measurement is a modification of the average square error, and is defined as

$$\text{Error} = \sum_{i=1}^{n} \frac{(y_i - y_i')^2}{2n},$$

where y_i is the correct output, y_i' is the output of the inference process using the rule set and n is the number of the examples.

Table 2
Results of different learning algorithms

	CART (%)	C4.5 (%)	C4.5rules (%)	SLAVE + R (%)
Monk1	83.27	75.70	100	100
Monk2	60.30	65.00	65.30	81.94
Monk3	92.32	97.20	96.88	97.20
IRIS	97.24	92.73	94.36	96.61

The refinement algorithm is a heuristic algorithm that attempts to maximize the good quality of the rules. The estimate of the good quality of the rule set is based on an error measurement. The algorithm uses the hill-climbing strategy and consists of the following four steps (see Fig. 8) that are repeated until the global error and the number of rules cannot be decreased (termination condition).

(1) *Improving the correctness of the rules:* When we work with problems where the variable consequent is discrete, the correctness is the relation between the number of examples classified and the number of examples covered by this rule. In the cases where the variable consequent is continuous, this concept must be extended because the output in the predictive process depends on the interaction between all the possible rules that are applicable. In the continuous case, we define the correctness of a rule as the average of the errors produced in the outputs in which this rule is used.

This step consists of a specification process where the irrelevant cases of the antecedent variables are eliminated for each rule. When these cases are eliminated from the description of the rule the correctness of the new rule is improved.

This part of the procedure is necessary for the SLAVE learning algorithm since its main problem is produced when the consistency parameters selected are not the most appropriate. In this situation, the correctness of the rules returned by the learning algorithm are too low and using this method, the correctness can be improved. Furthermore, any rule with a very low degree of correctness can be eliminated from the rule set. This is possible because the elimination of a rule is considered as a special case of specification (that is, when all the values of an antecedent variable are deactivated). The implementation of this process is similar to step 1 of the refinement of the crisp consequent domain.

(2) *Decreasing the error using a generalization process:* In this step, we try to improve the completeness of the rules using an order generalization process. The process consists in appending new cases for the antecedent variables if the global error is maintained or decreased. Its implementation is similar to step 2 of the refinement of the crisp consequent domain.

(3) *Decreasing the error by appending new rules:* This step has the same goal as the previous one but we are now interested in solving the problem by appending new rules. The main problem in this step is the following: which rule must we append to the rule set?

The answer to this question consists in establishing a criterion for deciding between all the possible rules that we can introduce. Given that the goal consists in finding the rule set that produces, on average, the smallest possible error on the training set, we introduce the most specific rule that reduces the largest error present in the training set. Therefore, the process consists in finding the example that has the maximum error and we will design a new rule from it.

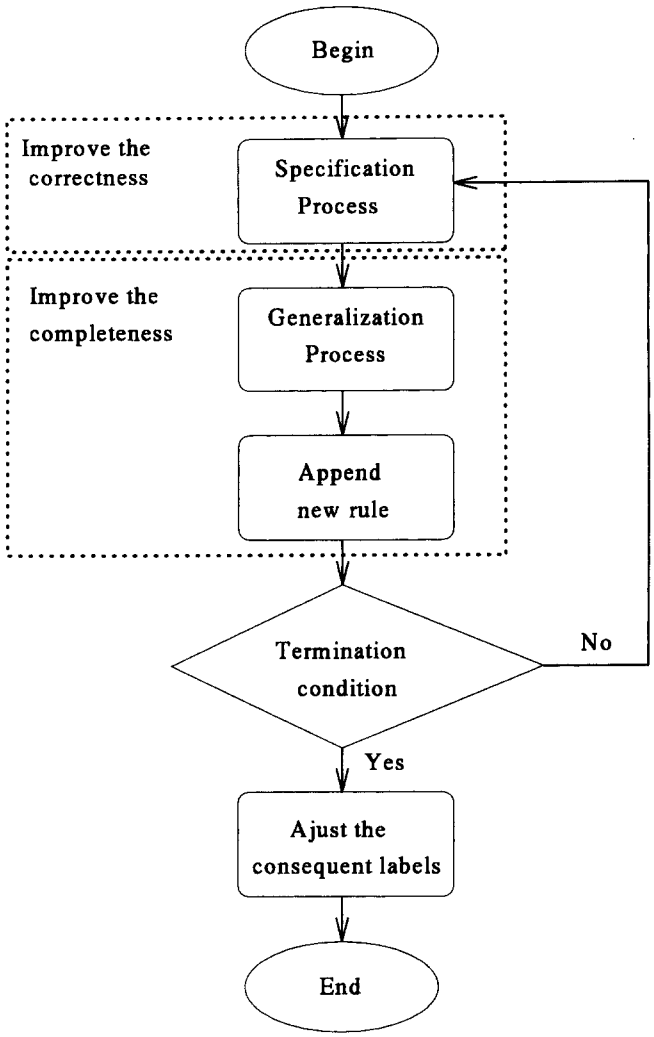

Fig. 8. Refinement for fuzzy consequent domain.

If there is more than one example with the maximum error, one of them is randomly selected. The new rule is made in the following way:

- Antecedent part: the most specific antecedent that covers this example.
- Consequent part: the label of the consequent that produces the smallest global error when this rule is included in the rule set.

The idea of this criterion is that the introduction of a rule that improves the maximum individual error must improve the average global error.

After the new rule is included in the rule set, it is generalized as far as possible following step 2 in this module and it is only applied for this rule. This step introduces rules until the last rule obtained can neither improve the maximum individual error nor the average global error. This last rule is not included in the rule set.

(4) *Adjusting the consequent labels:* In this part of the algorithm, we introduce a process for modifying the semantics of the consequent labels. In many cases, the semantic definition of the fuzzy labels is not well known. Therefore, the initial labels proposed can be considered as an approximation of the correct semantics. When we have a rule set that presents a description of the behavior of the system and we have an example set of this system, we can find a new semantic definition for the consequent labels that allows us to improve this description. In this way, we propose a simple heuristic process for modifying the semantics of trapezoid or triangular consequent labels. The process changes the width and the center points of each label, guided by the error measure in the following way.

1. Define a value δ that indicates the offset permitted in the labels.
2. For each label in the consequent variable carry out the following:
   2.1. The width of the label is altered in δ (first on the left and then on the right).
   2.2. If the error is strictly reduced, the modification is accepted.
   2.3. The center points of the label are altered in δ (first on the left and then on the right).
   2.4. If the error is strictly reduced, the modification is accepted.
3. If at least one change was accepted in step 2, then go to step 1. Otherwise, the process is finished.

4.1. An example of the behavior of the refinement

Let us suppose that SLAVE has learned the function

$$h(x,y) = x^2 + y^2$$

shown in Fig. 9(a), randomly extracting a set of examples for this function. Previously, a set of fuzzy labels on its domain was defined. Each domain is composed of seven fuzzy labels with triangular membership functions crossing at height 0.5 and distributed uniformly. The approximation obtained by SLAVE appears in Fig. 9(b) and we can see how SLAVE correctly approaches the central zone of the surface, where there are enough examples, but that error is very high in the extremes of the graph.

The refinement module attempts to correct the approximation obtained by SLAVE using the error measurement. In Fig. 9(c), the surface after the specification process is shown. This graph appears to present a worse approximation of the original because a strong slope appears on the surface, but by

taking the average error, this approximation is better. These strong slopes result from the elimination of bad rules which means that some of the examples are not well covered by the rule set.

From this step, we can see the development of the refinement algorithm as a process of smoother approximation on the original function. In Fig. 9(d), the control surface is improved by using the generalization process where the examples badly covered in the previous step, are covered to a better degree by the generalization of the existing rules. Fig. 9(e) shows the behavior of the rule set after step 3. The criterion used for introducing new rules consists in selecting a rule that reduces the maximum individual error. Using this criterion, we can see that the approximation of the extremes of the graph are more similar to the original graph.

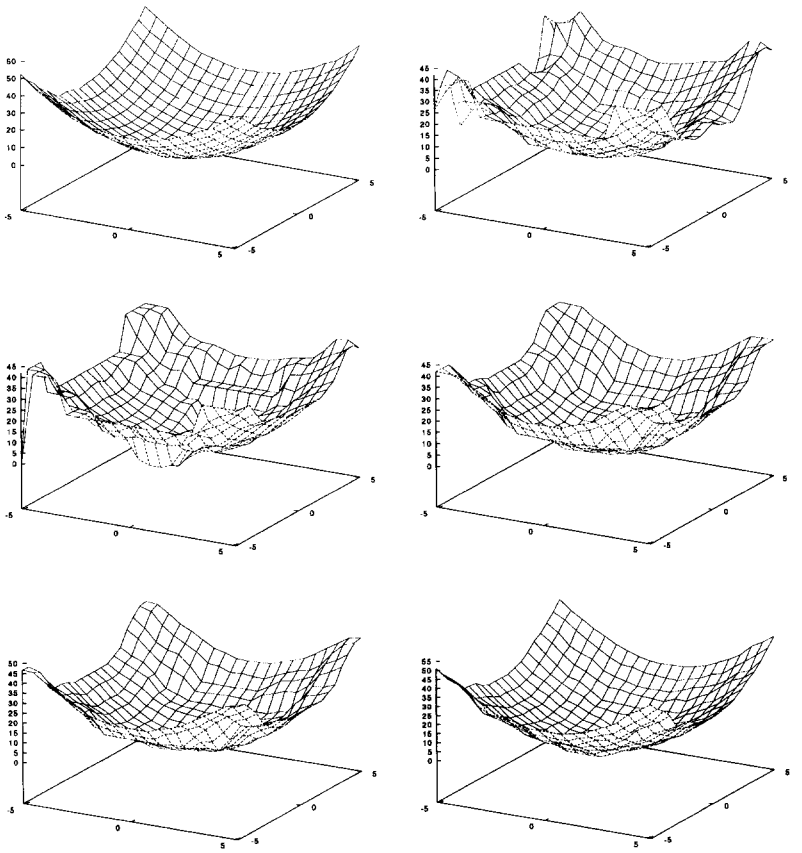

Fig. 9. An example of the refinement algorithm (a) Function $h(x, y) = x^2 + y^2$ (b) Approximation obtained by SLAVE (c) Approximation after step 1 (d) Approximation after step 2 (e) Approximation after step 3 (f) Approximation after refinement.

In order to solve this problem, the refinement algorithm is repeated eight times until the termination condition is satisfied. Fig. 9(f) shows the approximation returned by this module. This surface is very similar to the original surface and the improvement of the surface obtained by SLAVE is high.

The rule model of SLAVE is not usually used in problems with a fuzzy consequent domain. A rule model is normally used where each antecedent variable takes one and only one value of its domain [3,15,24]. In Section 2, we showed some of the advantages of this type of rule in that it allows us to determine the relevant variable for each concept and therefore it obtains more understandable descriptions. In Fig. 10, we propose two different non-linear functions to test if the refinement algorithm maintains this property. We have selected the function

$$f(x,y) = x^2$$

(Fig. 10(a1)) and SLAVE returns 11 rules for this problem (Fig. 10(b1)). SLAVE + R (Fig. 10(c1)) refines the rules of SLAVE and it returns six rules where the y variable is considered to be irrelevant for all the rules. Contrary to this function, the function

$$g(x,y) = \cos(x)\ \cos(y)$$

(Fig. 10(a2)) depends on two variables to determine the output. In this situation, SLAVE + R (Fig. 10(c2)) obtains a good approximation of the original g function, considering both predictive variables to be relevant for all of its rules.

4.2. *Applying SLAVE to a control problem*

The purpose of any controller is to periodically look at the values of the state variables in the controlled system and to obtain the values associated with their control variables by means of the relationships existing between them. If these relationships can be expressed in a mathematical way, it is not too difficult to design the controller. The problem comes when, as happens in a lot of real world non-linear systems with complex dynamics, there is no mathematical model representing the existing relationships.

Over the last few years, the application of Artificial Intelligence techniques has become a research topic in the domain of process control allowing efficient controllers to be obtained in cases in which a mathematical representation of the controlled systems cannot be obtained. Fuzzy Logic Control (FLC) is the main topic of this new field known as Expert Control. FLC was pioneered by Mamdani and Assilian in the work [17], and is now considered as one of the most important applications of Fuzzy Set Theory suggested by Zadeh [25]. Over the past few years, many applications of FLC have been developed successfully. Some of the most recent applications are water treatment, elevator

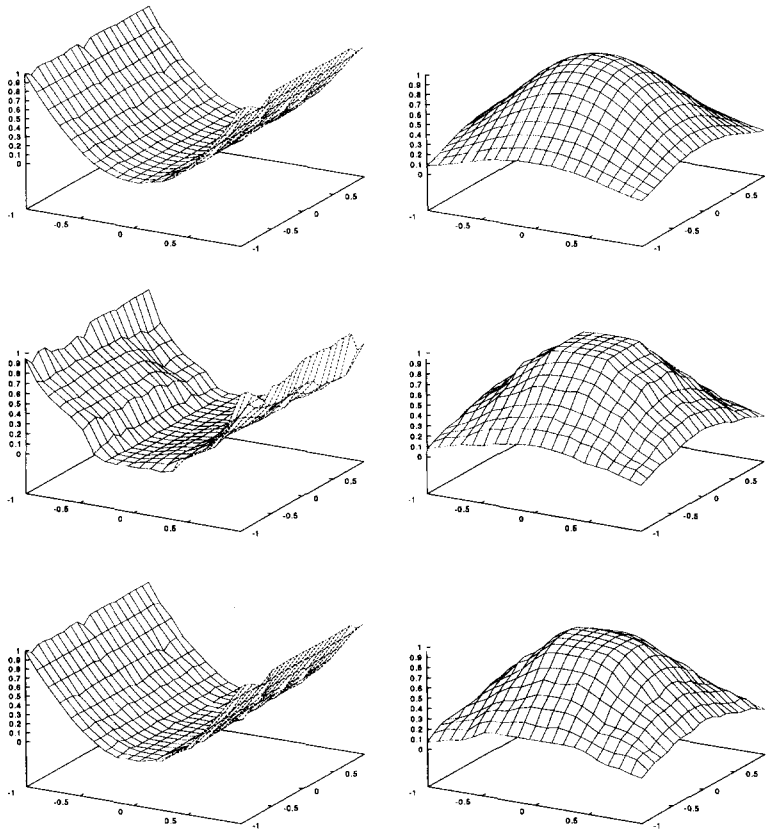

Fig. 10. An example of the refinement algorithm (a1) Function $f(x,y) = x^2$ (a2) Function $g(x,y) = \cos(x)\ \cos(y)$ (b1) Approximation of SLAVE (b2) Approximation of SLAVE (c1) Approximation of SLAVE + R (c2) Approximation of SLAVE + R.

control, video equipment, robot control and control of biological processes and several articles can be found about these topics [1,15,16].

A traditional control problem used for testing the behavior of any new controller is the pendulum problem. The pendulum problem consists in learning the rule set that permits the pendulum to be controlled, taking into account the example set that describes the system's performance. Assuming $|\theta| \ll 1$ (radian) the non-linear differential equation that leads to the behavior of the pendulum is managed by the equation

$$m\frac{l^2}{3}\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2} = \frac{l}{2}\left(-f + mg\ \sin\ \theta - k\frac{\mathrm{d}\theta}{\mathrm{d}t}\right)$$

where $kd\theta/dt$ is an approximation of the friction strength.

This system can be described using two state variables θ (angle) and ω (angular speed) and the control variable f (force). A pendulum weighing 5 kg and which is 5 m long has been considered in a real simulation, applying the force to the center of gravity, for a constant time of 10 ms. With these parameters, the discourse universe of the variables are the following:

$$\theta \in [-0.277, 0.277], \quad \omega \in [-0.458, 0.458], \quad f \in [-1593, 1593].$$

Using the previous restriction, we have experimentally obtained two example sets: the first one contains 213 examples and is used for learning (training set) and the second one contains 125 examples and is used for testing the behavior of the learned rule set (test set). These example sets have been obtained from the previous equation on two different initial conditions:

(a) $\theta = -0.277$ and $\omega = 0$.

(b) $\theta = 0.277$ and $\omega = 0$.

Fig. 11 represents the control surface on the previous conditions. The first step for working with the SLAVE learning system consists in discretizing the range of variables using fuzzy labels. Fig. 12 shows the labels used for each variable. Furthermore, for this problem, the value of δ has been simply defined as $(r_b - r_a)/100$, where r_b and r_a are the maximum and minimum value of the consequent variable respectively, that is $\delta = 59.6$.

We want to know the behavior of this refinement algorithm using different theories for this problem. Therefore, we propose the following knowledge bases:

UnK. This is an empty theory, that is, it has not contained any rule.

SLAVE. This is the knowledge base obtained by SLAVE.

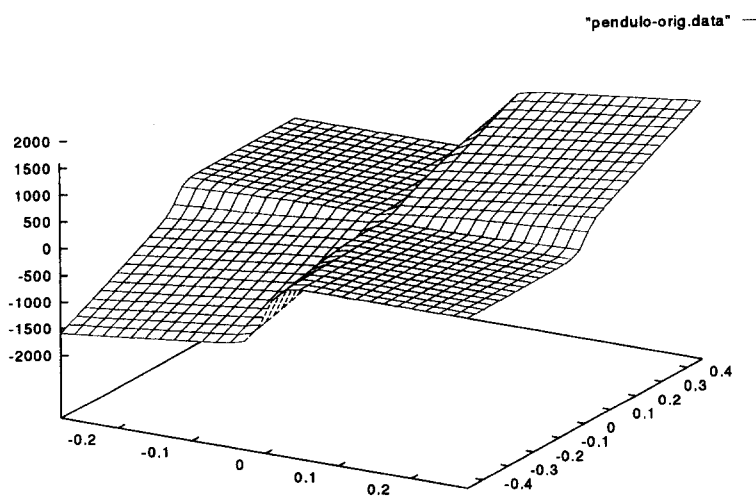

Fig. 11. Control surface of the pendulum problem.

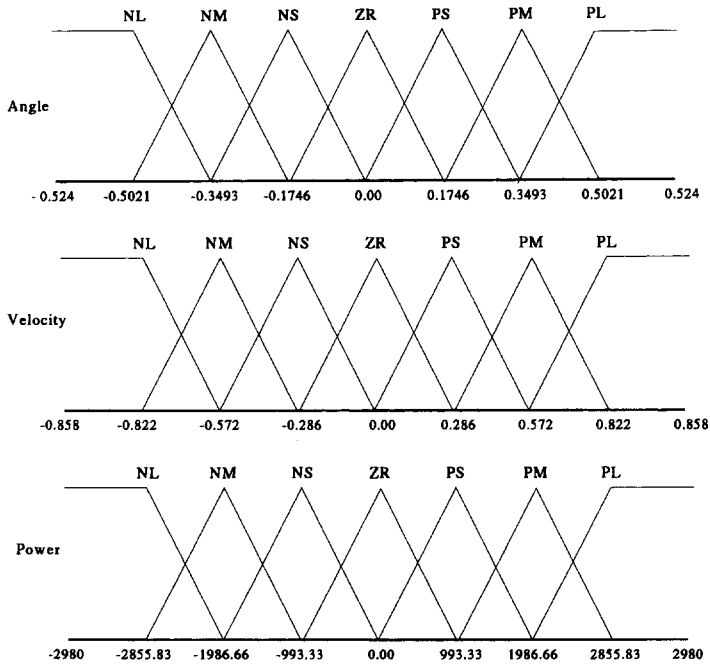

Fig. 12. Pendulum problem domains.

WM. This is the theory obtained using the Wang and Mendel algorithm [24].

For the experiment, we have studied the behavior of the previous knowledge bases and their combination with the refinement algorithm proposed in [11], that we will denote by +OldR, and with the refinement algorithm proposed in this work, that we denote +R, that is, for example SLAVE + OldR means

Table 3
Table of results

Rule database	Error	Number of rules
WM	7.45	9
SLAVE	492.16	6
UnK + OldR	216.01	9
WM + OldR	6.92	6
SLAVE + OldR	1.21	5
UnK + R	13.02	7
WM + R	0.14	5
SLAVE + R	0.14	5

the database obtained by the learning algorithm SLAVE and refined by the algorithm proposed in the previous work.

Table 3 shows the number of rules and the error obtained using the test example set with the most relevant combinations between the previous knowledge bases and the refinement algorithm. In all cases, we have taken the max–min inference system and the average of the center of gravity weighted by the matching value as a defuzzification method and the minimum operator as t-norm. The set of rules obtained by SLAVE + R is shown in Table 4.

Fig. 13 represents the control surface obtained by SLAVE and Fig. 14 the control surface obtained by SLAVE plus the refinement algorithm (by using the previous rule set). Fig. 15 shows that in the process of adjusting the labels of the consequent variable, the labels NM and PM are slightly modified in order to improve the behavior.

The Wang–Mendel result has been obtained from the learning algorithm proposed in [24] and using the same fuzzy discretization as we used in

Table 4
The fuzzy theory obtained by SLAVE + R

1 If θ is NS and ω is NS Then f is NS
2 If θ is PS and ω is PS Then f is PS
3 If ω is ZR Then f is ZR
4 If θ is NM or NS and ω is NM Then f is NM
5 If θ is PS or PM and ω is PM Then f is PM

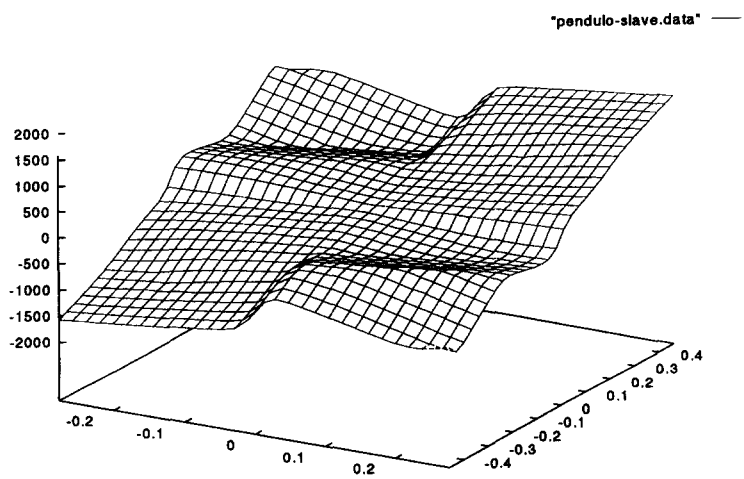

Fig. 13. Control surface obtained by SLAVE.

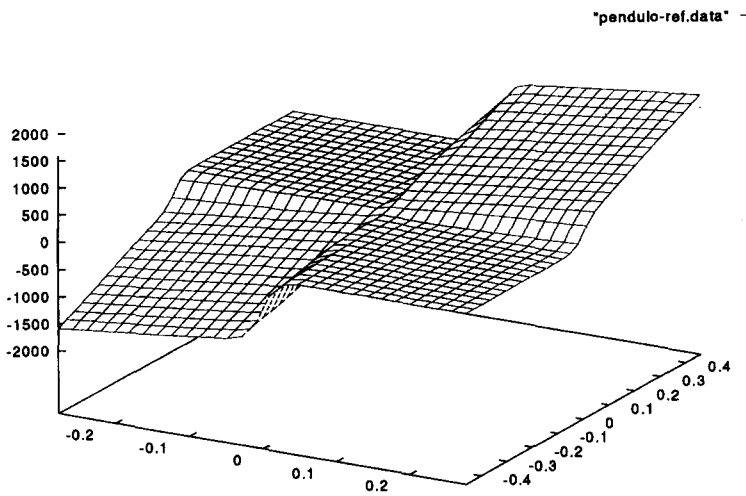

Fig. 14. Control surface obtained by SLAVE + R.

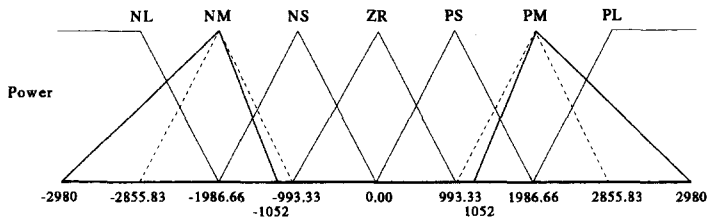

Fig. 15. Modification of the consequent variable labels.

SLAVE. From the results we can deduce the following conclusions for this problem.

(a) The refinement algorithm applied to UnK produces good results but they are not comparable with the results obtained by this refinement algorithm when the initial knowledge represents a good approximation of the system that we want to learn.

(b) The new proposal of the refinement algorithm has shown that better results are obtained than in the initial proposal.

(c) This algorithm of refinement, that was developed as a component of the SLAVE learning system, can be applied to knowledge bases obtained by other learning algorithms or obtained using other processes.

(d) SLAVE + R produces good results in accuracy, number of rules and the simplicity of the rules.

5. Conclusions

In this paper, we have described an algorithm for refining fuzzy rules. This algorithm uses a particular heuristic to combine the processes of specification, generalization, addition and elimination of rules. We have defined two different algorithms, one of them for problems where the consequent variable is crisp and the other for problems where the consequent variable is discretized using fuzzy labels. The objective in both cases is to improve the performance of the obtained knowledge and to improve the understanding of this knowledge.

This refinement algorithm can be applied to knowledge bases obtained from several sources, offering a significant improvement of this knowledge, but its development is strongly associated with the SLAVE learning system. The combination of both algorithms is called SLAVE + R. We have tested the behavior of SLAVE + R with respect to SLAVE and other learning algorithms in problems with crisp consequent variables and in problems with fuzzy consequent variables.

This learning system obtains a reasonably small set of fuzzy rules which can be easily understood from a human point of view. Finally, in this paper we have shown, that SLAVE + R produces good results when it is applied to different problems.

References

[1] P.P. Bonissone, Fuzzy logic controllers: An industrial reality, Computational Intelligence: Imitating Life, IEEE Press, New York, 1994, pp. 316–327.

[2] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, Classification and Regression Trees, Wadsworth, Belmont, CA, 1984.

[3] O. Cordón, F. Herrera, A three-stage process for learning descriptive and approximative fuzzy logic controller knowledge bases from examples, Internat. J. Approx. Reas. 17 (1997) 369–407.

[4] D. Fensel, M. Wiese, Refinement of rule sets with JoJo, Lectures Notes in Artificial Intelligence 667 (1993) 378–383.

[5] D. Driankov, H. Hellendoorn, M. Reinfrank, An Introduction to Fuzzy Control, Springer, Berlin, 1993.

[6] R.A. Fisher, The use of multiple measurements in taxonomic problems, Annual Eugenics 7 (1936) 179–188.

[7] A. González, R. Pérez, J.L. Verdegay, Learning the structure of a fuzzy rule: A genetic approach, Proceedings of the EUFIT'93, vol. 2, 1993, pp. 814–819; also Fuzzy System and Artificial Intelligence 3 (1) (1994) 57–70.

[8] A. González, R. Pérez, Structural learning of fuzzy rules from noisy examples, Proceedings of the FUZZIEEE/IFES'95, Yokohama, vol. III, 1995, pp. 1323–1330.

[9] A. González, R. Pérez, Completeness and Consistency conditions for learning fuzzy rules, Technical Report #DECSAI-95103, 1995 (to appear in Fuzzy Sets and Systems).

[10] A. González, R. Pérez, Refining the rules obtained by SLAVE, Technical Report #DECSAI-95138, 1995.

[11] A. González, R. Pérez, A learning system of fuzzy control rules, in: F. Herrera, J.L. Verdegay (Eds.), Genetic Algorithms and Soft Computing, Physica-Verlag, Wuezburg, 1996, pp. 202–225.
[12] A. González, R. Pérez, A refinement algorithm of fuzzy rules for classification problems, Proceedings of the IPMU'96, vol. 2, 1996, pp. 533–538.
[13] A. González, F. Herrera, Multi-stage genetic fuzzy systems based on the iterative rule learning approach (to appear in Mathware and Soft Computing).
[14] A. González, R. Pérez, Aplicación de un Sistema de Refinamiento de Reglas a Problemas de Clasificación, Memoria del V Congreso Iberoamericano de Inteligencia Artificial, Editorial Limusa, 1996, pp. 20–29.
[15] H. Hellendoorn, D. Driankow, M. Reinfrank, An introduction to Fuzzy Control, Springer, Berlin, 1993.
[16] C.C. Lee, Fuzzy logic in control systems: Fuzzy logic controller, Parts I and II, IEEE Transactions on Systems, Man and Cybernetics 20 (1990) 404–435.
[17] E.H. Mamdani, S. Assilian, An experiment in linguistic synthesis with a fuzzy logic controller, International Journal of Man–Machine Studies 7 (1975) 1–13.
[18] R.S. Michalski, A theory and methodology of inductive reasoning, in: R.S. Michalski, J. Carbonell, T. Mitchel (Eds.), Machine Learning: An artificial intelligence approach, vol. 1, Morgan Kaufmann, Los Altos, CA, 1984, pp. 83–134.
[19] D. Ourston, R.J. Mooney, Theory refinement combining analytical and empirical methods, Artificial Intelligence 66 (1994) 273–309.
[20] J.R. Quinlan, Induction of decision trees, Machine learning 1 (1) (1986) 81–106.
[21] J.R. Quinlan, C4.5 Program for Machine Learning, Morgan Kaufmann, Los Altos, CA, 1993.
[22] S.B. Thrun et al., The monk's problems: A performance comparison of different learning algorithms, Technical Report CMU-CS-91-197, Computer Science Department, Carnegi Mellon University, Pittsburgh, 1991.
[23] G. Venturini, SIA: a Supervised Inductive Algorithm with Genetic Search for Learning Attributes based Concepts, Machine Learning: ECML-93, 1993, pp. 280–296.
[24] L. Wang, M. Mendel, Generating fuzzy rules by learning from examples, IEEE Transactions on systems, man and cybernetics 22 (6) (1992) 1414–1427.
[25] L.A. Zadeh, Fuzzy set, Information and Control 8 (1965) 338–353.