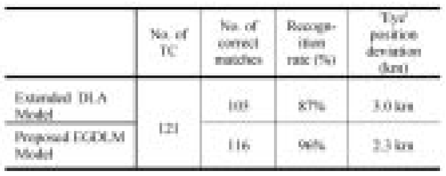
TABLE II
TC PATTERN RECOGNITION RESULT FOR 120 EIR SATELLITE IMAGE CASES



	No. of TC	No. of correct matches	Recognition rate (%)	'Eye' position deviation (km)
Extended DLA Model	121	103	87%	3.0 km
Proposed EGDLM Model		116	96%	2.3 km

the Elastic Graph Dynamic Link Model (EGDLM). Results are challenging, with an overall correct recognition rate of more than 95%.

The promising results can be explained in two ways. First, the Dvorak technique provides a high precision and scientific scheme for TC pattern classification and identification through pattern matching technique [5], [6]. However, due to the high variation of cloud patterns in satellite pictures, subjective human vision matching is being used so far by trained meteorological analysts. By using EGDLM, the human matching problem can be automated by a computer matching process using the "elastic graph matching" scheme. Second, another vital obstacle for the automation of the satellite interpretation task is the problem of scene analysis. In the paper, the active contour model (ACM) provides a feasible and efficient solution by "contour" extraction to "isolate" the cloud systems into groups which will efficiently reduce the tedious time-consuming matching process.

Another vital finding is to make use of both visible and EIR images to improve the recognition rate. By using the EIR satellite images in the recognition process, the vorticity contour features of the TC cloud bands for different temperatures can be made easily distinguishable and the recognition rate is improved. Nevertheless, visible images still play a vital role, as the main function of scene analysis in satellite interpretation is to "extract" the weather system for further analysis, which also resembles the human vision-processing scheme.

ACKNOWLEDGMENT

The authors wish to thank NOAA for the provision of satellite pictures (both visible and EIR) for TC cases from 1990 to 1998.

REFERENCES

[1] E. Bienenstock and C. von der Malsburg, "A neural network for invariant pattern recognition," *Europhys. Lett.*, no. 4, pp. 121–126, 1987.
[2] V. Caselles *et al.*, "Geodesic active contours," *Int. J. Comput. Vision*, vol. 22, no. 1, pp. 61–79, 1997.
[3] L. D. Cohen, "NOTE on active contour models and balloons," *CVGIP: Image Understanding*, vol. 53, no. 2, pp. 211–218, 1991.
[4] T. F. Cootes *et al.*, "Multi-resolution search with active shape models," *Proc. IEEE Int. Conf. Comput. Vision Image Process.*, pp. 610–612, 1994.
[5] V. F. Dvorak, "A technique for the analysis and forecasting of tropical cyclone intensities from satellite pictures," U.S. Dept. Commerce, Washington, D.C., NOAA Tech. Memo. NESS 45, 1973.
[6] ——, "Tropical cyclone intensity analysis and forecasting from satellite imagery," Mon. Weather Rev., no. 103, 1975.
[7] ——, "Tropical cyclone intensity analysis using satellite data," NOAA Tech. Rep., Washington, DC, NESDIS 11, 1984.
[8] R. Goldenberg *et al.*, "Fast geodesic active contours," in *Scale-Space Theories in Computer Vision.* ser. Lecture Notes in Computer Science 1682, M. Nielsen *et al.*, Eds. New York: Springer-Verlag, 1999, pp. 34–45.
[9] M. Kass and A. Witkin, "Snakes: Active contour models," in *Proc. Int. Conf. Comput. Vision*, 1987, pp. 259–268.
[10] N. Kruger, "An algorithm for the learning of weights in discrimination functions using prior constant," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, pp. 764–768, July 1997.
[11] M. Lades, "Distortion invariant object recognition in the dynamic link architecture," *IEEE Trans. Comput.*, vol. 42, pp. 300–311, Mar. 1993.
[12] R. S. T. Lee and J. N. K. Liu, "An automatic satellite interpretation of tropical cyclone patterns using elastic graph dynamic link model," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 13, no. 8, pp. 1251–1270, 1999.
[13] F. Leymarie and M. D. Levine, "Tracking deformable objects in the plane using an active contour model," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 15, pp. 617–634, June 1993.
[14] J. N. K. Liu and R. S. T. Lee, "Invariant character recognition in dynamic link architecture," in *Proc. KDEX*, Newport Beach, CA, 1997, pp. 188–195.
[15] J. N. K. Liu and R. S. Lee, "Invariant handwritten Chinese character recognition by dynamic link architecture," in *Proc. ICONIP/JNNS*, vol. 1, Kitakyushu, Japan, pp. 275–278.
[16] C. von der Malsburg, "The correlation theory of brain theory," MPI Biophys. Chem., Int. Rep. 81-2, 1981.
[17] ——, "Nervous structures with dynamical link," *Ber. Bunsenges. Phys. Chem.*, vol. 87, pp. 703–710, 1985.
[18] W. Neuenschwander *et al.*, "Initializing snakes," in *Proc. CVPR*, Seattle, WA, 1994, pp. 658–663.
[19] C. S. Velden *et al.*, "Tropical cyclone center-fixing using DMSP SSM/I data," in *Proc. Fourth Conf. Satellite Meteorol.*, San Diego, CA, 1989, pp. J36–J39.
[20] C. S. Velden *et al.*, "Objective Dvorak technique (ODT)," Weather and forecasting, regional and mesoscale meteorology branch, NOAA/NESDIS, U.S. Dept. Commerce, Washington, DC, Mar. 1998.
[21] L. Wiskott and C. von der Malsburg, "Recognizing faces by dynamic link matching," in *Proc. ICANN*, Paris, France, pp. 347–352.
[22] R. P. Wurtz, "Object recognition robust under translations, deformations and changes in background," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, pp. 769–775, July 1997.

Selection of Relevant Features in a Fuzzy Genetic Learning Algorithm

Antonio González and Raúl Pérez

***Abstract*—Genetic algorithms offer a powerful search method for a variety of learning tasks, and there are different approaches in which they have been applied to learning processes. Structural learning algorithm on vague environment (SLAVE) is a genetic learning algorithm that uses the iterative approach to learn fuzzy rules. SLAVE can select the relevant features of the domain, but when working with large databases the search space is too large and the running time can sometimes be excessive. We propose to improve SLAVE by including a feature selection model in which the genetic algorithm works with individuals (representing individual rules) composed of two structures: one structure representing the relevance status of the involved variables in the rule, the other one representing the assignments variable/value. For this general representation, we study two alternatives depending on the information coded in the first structure. When compared with the initial algorithm, this new approach of SLAVE reduces the number of rules, simplifies the structure of the rules and improves the total accuracy.**

***Index Terms*—Feature selection, fuzzy rules, genetic algorithms (GAs), machine learning.**

I. INTRODUCTION

Genetic algorithms (GAs) are search algorithms that use operations found in natural genetics to guide the trek through a search space. GAs have been theoretically and empirically proven to provide robust search

Manuscript received May 12, 1998; revised February 3, 2001. This work was supported by CICYT under Grant TAP99-0535-C02-01. This paper was recommended by Associate Editor L. O. Hall.

The authors are with the Departamento de Ciencias de la Computacióne e Inteligencia Artificial, ETS de Ingeniería Informática, Universidad de Granada, Granada, Spain (e-mail: a.gonzalez@decsai.ugr.es; fgr@decsai.ugr.es).

Publisher Item Identifier S 1083-4419(01)04857-9.

capabilities in complex spaces, offering a valid approach to problems requiring efficient and effective searching. This is the case in many problems of learning attribute-valued rules.

The most well-known approaches in which GAs have been applied to learning processes are the Michigan [19] and the Pittsburgh [34] approaches. In the first of these, each chromosome corresponds to only one rule and the population represents the complete set of rules, whereas in the Pittsburgh approach, each chromosome encodes a complete set of rules. In the first case, the solution is the complete population and in the second one, only the best chromosome is a solution for the problem [7]. A third way has been presented as an alternative to these models: the iterative approach [13], [17] where each chromosome represents only one rule but now the population does not represent the complete set of rules since this set is obtained by different runnings of the GA.

The combination of GA and soft computing in order to develop learning algorithms is particularly interesting. See, for example, [4], [6], [28], and [29], where different algorithms for fuzzy logic control are presented, and [11], [20] which proposes algorithms for classification problems.

An example of this combination is the structural learning algorithm on vague environment (SLAVE) [15], [16], which is a genetic learning algorithm previously developed by the authors that uses the iterative approach and a fuzzy description of the rules and examples. SLAVE uses an attribute-value language based on the use of linguistic variables [38]. Since SLAVE is based on the genetic iterative approach, it learns only one fuzzy rule in each execution of the GA. It fixes a class and selects the best antecedent for this class. The complete set of rules is obtained through a sequence of repeated executions of the GA.

The selection of relevant features, and the elimination of irrelevant ones, is a very important point for many learning problems. In several problems the number of possible variables to be considered is very large, and in this case, the search space for the learning algorithm is big enough. In order to solve many practical problems, it is therefore essential to consider the feature selection in a learning algorithm.

Because of the particular type of rule used in SLAVE, the irrelevant variables in a rule can be eliminated during the learning process. The computational cost of eliminating a variable in a rule, however, is high, and we therefore want to modify SLAVE by including a powerful mechanism for feature selection. The idea is to modify the genetic algorithm of SLAVE so that it can learn efficiently when dealing with problems in which the training examples have a high number of variables and/or a high number of values in each variable. In the literature on this subject, different approaches have been proposed which combine feature selection and GA, such as [3], [22], [27], [33], and [35]. In our case, we are interested in designing an embedded model of feature selection [26] for SLAVE; and in particular, an embedded model in which the feature selection is made for each particular rule (because of the iterative process followed in SLAVE), i.e., the learning algorithm must select the appropriate set of variables and values for each particular rule. In this process, each rule (with a particular class value) may have a different subset of variables to identify the class.

In order to include the feature selection in the genetic learning algorithm of SLAVE, the main idea is to use a genetic representation where each individual is composed of two structures: one structure codes the relevance of the predictive variables involved in the learning problem and the other one codes the assignments variable/value. Consequently, each one of the two structures has a different task associated inside the rule selection module. The first structure (called the variable chromosome) attempts to find the most appropriate set of features for the antecedent of the rule, while the second one (called the value chromosome) attempts to find for each predictive variable the most appropriate assignment of values from its domain. The genetic algorithm simultaneously carries out both tasks during the evolutive process. Furthermore, each structure has its own set of genetic operators.

This new genetic representation establishes a general framework with different possible alternatives and in this paper, we explore some of them. In the first approach, we encode the relevance of each variable (variable chromosome) through a zero-one array, with one indicating that the variable is relevant and with zero that it is not relevant. Although this initial alternative significantly improves the behavior of the original SLAVE, it does not take advantage of the information given by the distribution of examples in the training set. This information can be used to establish the degree of relevance of each variable to determine a class in a similar way to the generation of decision trees [32]. In the second approach, we explore the use of this information to improve the feature selection mechanism.

In the following section, we provide a general description of the basic SLAVE algorithm. Section III describes the main characteristics of the new genetic learning algorithm and explains how this new version of SLAVE can select the relevant variables in a problem. Section IV explores the use of information about the relevance degree of each variable. Finally, the last section shows the behavior of the different proposed alternatives on several problems.

II. General Description of Slave

GAs have been used widely to develop learning algorithms. The well-known Michigan and Pittsburgh approaches have been the basis for learning algorithms such as the classifier systems [18], GABIL [9], or GIL [21]. An alternative to these is the iterative approach described in [13] and [17] and used in systems like SLAVE [15], [16], SIA [36], or MOGUL [5].

The iterative approach attempts to reduce the search space of possible solutions by searching for only one rule at a time. The main idea is to reduce the original problem of obtaining a complete set of rules to a simpler problem which consists in obtaining only one rule at a time. In this approach, as in the Michigan one, each chromosome in the population represents a single rule, but now, only the best individual is considered, and the remaining chromosomes in the population are discarded. Therefore, in the iterative model, one execution of the GA provides a partial solution (a rule) to the learning problem. In order to obtain a set of rules, which will be a complete solution to the problem, the GA must be placed within an iterative scheme similar to the following.

1) Use a GA to obtain ONE RULE for the system.
2) Incorporate the rule into the final set of rules.
3) Penalize this rule.
4) If the current set of rules adequately represents the examples in the training set, the system returns the set of rules as the solution. Otherwise, go to step 1.

A very easy way to penalize the rules already obtained (step 3), and thus be able to learn new rules, consists in eliminating from the training set all those examples that are covered by the set of rules previously obtained. When all the examples have been eliminated, the algorithm has detected an adequate set of rules. This process does not need to fix a priori the number of rules or the number of necessary GA runs in order to obtain the final set of rules.

SLAVE [15], [16] is an inductive learning algorithm based on the iterative approach. SLAVE uses a set valued model of the rule

$$IF\ X_1\ is\ A_1\ and\ \cdots\ and\ X_n\ is\ A_n\ \ THEN\ \ Y\ is\ B$$

where the value assigned to each antecedent variable, i.e., A_i, is allowed to be a subset of simple fuzzy values of the domain. For example, let AGE be the variable with fuzzy domain

$$D = \{\text{YOUNG, YOUNG-ADULT, ADULT, OLD, VERY-OLD}\}$$

and the semantics of these labels those described in Fig. 1. An assignment to the AGE variable such as

$$AGE = \{\text{YOUNG, YOUNG-ADULT, ADULT}\}$$

is equivalent to

$$\{\text{AGE is YOUNG or AGE is YOUNG-ADULT or AGE is ADULT}\}.$$

Using the formulation proposed in [15], where the disjunction of adjacent values is considered to be the convex hull of the fuzzy labels, the previous assignment is equivalent to

$$\{\text{AGE is less than or equal to ADULT}\}.$$

With this rule model and using the formulation described in [15], when the value of a variable coincides with its whole domain, the variable is irrelevant for describing the particular class in the rule and may be eliminated.

With respect to the implementation of the fuzzy if–then system, the inference model used for these rules is similar to the models used for rules with simple values. Thus, given an example $e = (e_1, e_2, \ldots, e_n)$, the associated class for this example is that which corresponds to the rule with the biggest match between the example and the antecedent of the rule. In order to calculate a measure of this match, we use a t-norm to combine the partial matches between e_i and A_i. Since A_i is now a subset of fuzzy values, this match was defined in [15] using a normalized maximum operator on the different values A_i. When these values are consecutive, this operator is equivalent to calculating the membership function of e_i on the convex hull of the different values of A_i.

Taking into account this rule model and its associated inference process, SLAVE selects a set of rules that describe a training set. The most important component of SLAVE is the rule selection process. This consists in obtaining the best rule in each execution of the GA depending on the examples of the training set. The concept of the best rule is based on the notion of consistency and completeness. We propose a degree of completeness and a degree of consistency for a rule where both definitions use the concept of number of positive and negative examples for a rule defined in [15] and improved in [16]. The basic idea is to use the fuzzy cardinal of fuzzy sets "positive examples" and "negative examples" for a rule. The membership function of an example to the fuzzy set "positive examples" is defined by combining the match between the antecedent and the example and the match between the value of the class of the rule and the class of the example. The membership function of an example to the fuzzy set "negative examples" uses the same matching measure for antecedents, but for the consequent it computes the match between the class of the example and the set of values of all consequent variables except that of the value of the class of the rule. A complete description can be found in the references [15] and [16].

Definition 1: The degree of completeness of a rule R is defined as

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

where $n^+(R)$ is the number of positive examples to the rule R and n_B is the number of examples in the training set of the class B, with this number being calculated by using the membership function μ_B.

The soft consistency degree [15] is based on the possibility of admitting some noise in the rules. Thus, in order to define this degree, we use the following set:

$$\Delta^k = \{R | n^-(R) < k \times n^+(R)\}$$

where $\times$ is the product operator and $k \in [0, 1]$. This last equation represents the set of rules having a number of negative examples $n^-(R)$ strictly less than a fraction k of the positive examples $n^+(R)$.

Definition 2: The degree to which a rule with $n^+(R) > 0$ satisfies the soft consistency condition is

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

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

Definition 2 uses two parameters: k_1 is a lower bound of the noise threshold and k_2 is an upper bound of the noise threshold.

The iterative approach of SLAVE fixes a class and the GA selects a rule that simultaneously verifies the completeness and the soft consistency condition to a high degree. The rule selection in SLAVE can therefore be solved by the following optimization problem:

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

where $D = P(D_1) \times P(D_2) \times \cdots \times P(D_n)$ with D_i being the fuzzy domain of X_i variable, $R_B(A)$ represents a rule with antecedent value $A = (A_1, \ldots, A_n) \in D$ and consequent value B, with B being fixed in the optimization problem. The iterative approach will change this consequent value to obtain the different values.

Details of the GA used in this optimization process can be found in [14]. The GA of SLAVE can eliminate a variable in a rule by chance when the complete domain D_i is assigned to a variable, but this process is neither systematic nor efficient. In the next section, we propose a modification of the GA of SLAVE that includes a selection of relevant features for each learned rule.

III. Proposal for the Inclusion of a Feature Selection in SLAVE

The main component of the SLAVE learning algorithm is the module of rule extraction. Once a class has been fixed, a GA attempts to find the best combination of value assignment to the antecedent variables of a rule within the class.

The new GA for SLAVE is described by means of the following five components:

1) a genetic representation of solutions to the problem;
2) a way to create an initial population of solutions;
3) an evaluation function which gives the fitness of each chromosome;
4) a set of genetic operators;
5) a termination condition for the genetic algorithm.

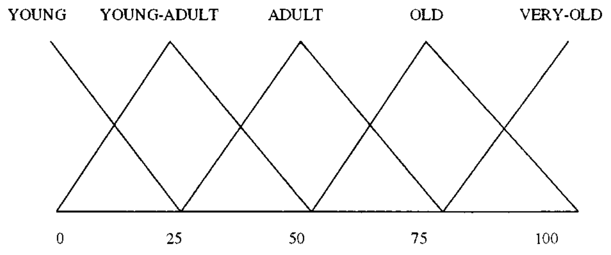


Fig. 1. Fuzzy domain of Example 1.

A. Representing the Population

The most important change with respect to the original version of SLAVE is the representation of the population, since it must allow us to obtain the relevant variables for a particular rule easily.

The representation of the genetic population in the original SLAVE contains a set of candidate antecedents for each class. Each candidate antecedent (i.e., each individual in the population) is composed of the sequence of assignments to the predictive variables, where each assignment variable/value is represented by a binary string. An individual of the population is composed by concatenation of the binary strings that represent the assignments variable/value on each predictive variable.

For example, the representation of an AGE variable (with the domain shown in Fig. 1) needs a binary string with five components (one for each label), where each component represents the absence or presence of the label in the assignment of the variable. The binary string **10 010** associated to the AGE variable, represents the assignment $AGE = \{YOUNG, OLD\}$, and the binary string **11 111** represents the irrelevance of the AGE variable (since the convex hull of all the labels of the domain is equivalent to the complete domain).

Let us suppose that AGE is a predictive variable involved in a learning problem, and that AGE is an irrelevant variable for the current class. Assuming that an individual of the population has the correct assignments on all variables except AGE, which has the current assignment $AGE = \{OLD\}$, i.e., the binary string **00010.** In order to obtain the best antecedent in this case, the genetic operators should transform the assignment of AGE from **00 010** to **11 111.** This transformation toward the correct configuration of the antecedent requires that the genetic operators do not affect the rest of the variables but make the correct change in the AGE variable. Therefore, when irrelevant variables appear, the GA of SLAVE has a high computational cost in the process of obtaining the best antecedent.

In general, the task of changing relevant variables to irrelevant variables in the genetic algorithm depends on the size of the binary coding of each variable and the number of predictive variables involved in the learning problem. We can say that the genetic algorithm of SLAVE eliminates irrelevant variables. However, with the current genetic representation of the information it is more difficult to eliminate a variable than to add it, and the system might take a long time to find a good solution.

Consequently, if we want to improve the detection of the irrelevant variables, we first need to change the genetic representation. So, our goal consists in obtaining a better representation of the genetic solutions to make a feature selection for each rule possible.

The idea is to include a new binary value associated to each antecedent variable in order to discover if the variable will be considered as part of the antecedent of the rule or not. The representation of an individual in the genetic population of the AGE example is encoded using a binary string with $5+1$ components, where one of these values determines the relevance or irrelevance of the variable, and the others determine the values assigned to the variable if it is considered to be relevant. With this new coding, a simple mutation on the new zero-one component changes its relevance status. This coding therefore reduces the computational cost associated with the detection of the irrelevant variables.

By considering this process for all the antecedent variables, we have two structures to represent the complete antecedent of a rule (see Fig. 2): One codifies the relevance of the variables and the other codifies the assignments variable/value. With this decomposition, the GA representation has a complex chromosome composed of two structures: a variable chromosome and a value chromosome. This division allows us to clearly distinguish between the two different tasks that are simultaneously carried out in the genetic algorithm (search for the appropriate variables and search for the appropriate value assignments) and we can associate the most appropriate set of genetic operators and set these operators on each structure.

We can now define both structures precisely. Let us suppose we have n possible antecedent variables $X_1, \ldots, X_n$ each X_i having an associated fuzzy domain D_i with m_i components. The genetic code needs information about the relevance of the variables and the value of these variables. The genetic code therefore has the following two structures:

- a variable chromosome;
- a value chromosome.

The variable chromosome (`VAR`) consists of a binary array of length n (the number of variables) with 1 indicating that the variable is relevant (or is active) and 0 indicating that the variable is irrelevant (or is inactive). The value chromosome (`VAL`) keeps the same binary coding as in SLAVE [17] and which was described above. In order to encode any elements of $P(D_1)\times\cdots\times P(D_n)$ we use a vector of $m_1+\cdots+m_n$ zero-one components (active or inactive values), with

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

with $s \in \{1 \cdots m_r\}$ and $r \in \{1 \cdots n\}$.

Example 1: Let us suppose that we have three variables, X_1, X_2, and X_3, with the associated fuzzy domain shown in Fig. 3. In this case, the code

$$\texttt{VAR}\ 110\ \texttt{VAL}\ ((111)(11000)(10))$$

represents the following antecedent:

$$X_1 \text{ is } \{A_{11}, A_{12}, A_{13}\} \text{ and } X_2 \text{ is } \{A_{21}, A_{22}\}.$$

Since X_1 ranges over all the elements in the domain D_1, the antecedent above is equivalent to

$$X_2 \text{ is } \{A_{21}, A_{22}\}$$

or alternatively, if the referential set of variable X_2 is an ordered set, the previous antecedent is equivalent to

$$X_2 \text{ is less than or equal to } A_{22}.$$

The variable X_3 was eliminated from the antecedent by the variable chromosome since it was irrelevant, but the variable X_1 was eliminated by the chromosome value as it covers all the domain and we use a normalized maximum operator equivalent to the convex hull of the different labels.

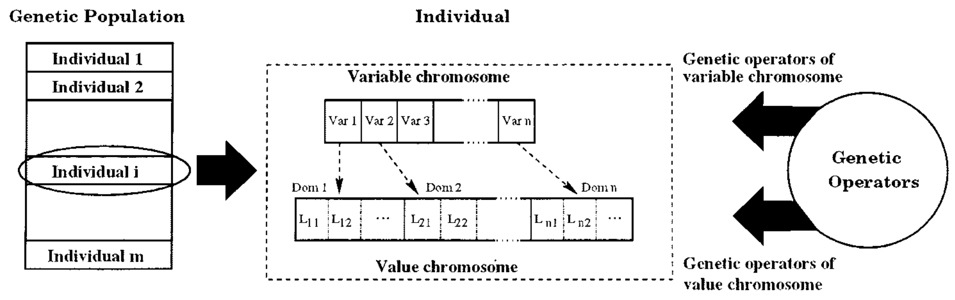


Fig. 2. Genetic population with two structures in each individual.

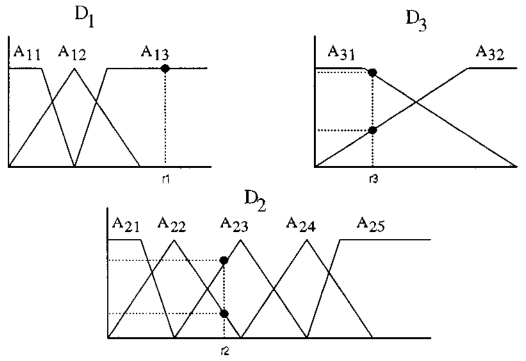


Fig. 3. Domains of variables X_1, X_2, X_3 in Examples 1, 2, and 3.

B. Generating the First Population

We define a different procedure to generate the initial information in each structure. The variable chromosome is built using an *initial activation probability* $p \in [0, 1]$. The meaning of p is the probability of considering a variable as relevant. This probability is only used to generate the initial values and it is the population evolution which must configure the most convenient structure for the antecedent. In the experiments, we will use an activation probability $p = 0.5$. The value chromosome is obtained by selecting examples of the training set at random from the class that must be learned and by assigning the most specific antecedent that best covers it. This antecedent consists of only one label for each antecedent variable. The label for each antecedent variable is the one that gives the highest degree of membership for each component in the example.

Example 2: Let X_1, X_2 and X_3 be the variables with the associated domain shown in Fig. 3, and let (r1, r2, r3) be the randomly selected example from the training set of a class. The most specific antecedent that best represents this example is

$$X_1 \ is \ A_{13} \ and \ X_2 \ is \ A_{23} \ and \ X_3 \ is \ A_{31}$$

with the binary representation (001)(00100)(10).

Using 0.5 as the initial activation probability, let us suppose that we obtain 101 for the variable chromosome, so the genetic code is

$$\texttt{VAR}\ 101\ \texttt{VAL}\ ((001)(00100)(10)).$$

The process is repeated to initialize each individual of the genetic population, selecting examples from the training set at random.

C. Evaluation Function (Fitness)

The aim of the GA is to find the best rule. The best rule is defined here as the one which simultaneously has the highest degrees of consistency and completeness and combines both factors using a product operator. Using the previous definitions for a rule R and a set of training examples E, we can define the evaluation function as

$$fitness(R) = \Lambda(R) \times \Gamma_{k_1 k_2}(R).$$

D. Genetic Operators

Both structures of the GA use an elitist model, the calculation of the selection probabilities follows a linear ranking [1] and the sampling algorithm is the roulette wheel selection [18]. The variable and value chromosome uses the following well-known standard operators [8], [18]:

- two point crossover operator;
- uniform mutation operator.

Moreover, in order to increase the diversity of the population, two additional operators have been considered for the value chromosome:

- the AND and OR operators;
- the rotation operator.

The AND and OR operators exchange genetic information in a similar way to the crossover operator. These operators try to simulate the behavior of the classical operators of generalization and specialization. The performance of these operators is as follows: Two points from two different chromosomes are selected at random and the two segments of the genetic code are combined using the AND operator or the OR operator (both of them gene to gene). The transformation only modifies the first parent selected, which is replaced by the offspring generated. In a similar way to the crossover operator, the processes on the variable and value levels are carried out independently. The rotation operator [14] is a modified version of the traditional inversion operator that we proposed in order to include a higher diversity in the searching problem. This operator takes a cutoff point in an element of the population and interchanges the position of the two segments.

E. Termination Condition

The GA must finish the process of searching for the best rule when we have a good reason to believe that this rule has been obtained given the current training set. In the implementation of the termination condition, we distinguish between the extraction of rules from a class in which we have at least one learned rule and a class in which we have no rules. We make a wider search when we wish to find the first rule of a class and relax this search process when we already have some rules for a class. The GA will return the best rule from the last population if one of the following conditions is satisfied.

- The number of iterations is greater than a fixed limit.
- The fitness value of the best rule in the population does not increase for at least a fixed number of iterations and rules with this consequent have already been obtained in previous runs.
- No rules with this value of the consequent have previously been obtained; but the fitness value does not increase for a fixed

number of iterations and the current best rule can eliminate at least one example from the training set.

The complete learning algorithm that uses a genetic algorithm based on the representation described in this section will be called 2SLAVE. The experiments performed will be presented in Section V. However, the variable chromosome is initially generated randomly without taking into account the information provided by the training set. To account for this information, it is necessary to investigate an alternative assignment for the initial code in the variable chromosome. The main idea is to replace the binary code (active or inactive) by a real code reflecting the relation between each variable and the class of the training examples.

IV. Using Information in the First Population

The initial code of a chromosome variable for each individual has been assigned through a default activation probability. This initialization assumes that we have no information about the relevance of the variables. The aim of this section is to find a way to assign the initial values of relevance for each predictive variable in the first population depending on the information provided by the training set.

A. Information Measures

The idea is to use a measure of the relevance of each variable with respect to a particular concept. At the beginning of the process, this measure will determine the capacity of the variable to represent the values of each class. Later on, the genetic process, using a set of operators, will modify these values and will therefore modify the criterion to determine the set of variables that must be activated to represent each class.

Given the variables X and Y, we use the following information measure [25], [37]

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

where x and y are particular values of the variable X and Y, respectively, the sum ranges over the values of the domain of each variable and $p()$ is a probability measure.

This measure defines a different range for each pair of variables. In order to facilitate the comparison of the results in all the variables, it is necessary for all measures to be in the same range. In order to achieve this, we use the following normalization:

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

where $H(X, Y)$ 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 τ measure estimates the dependence between variables X and Y in the following way: Values of $\tau(X, Y)$ 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 the learning algorithm SLAVE, we have selected rules fixing a class of the consequent variable. We therefore need to restrict the previous measure for the particular class that is being learned, and we define

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

where

X	predictive variable;
Y	classification variable;
C	particular class.

Like $\tau(X, Y)$, $\tau_C(X)$ measures the dependence or independence degree between the X variable and the C value of the consequent variable. We interpret this value as the relevance value of each X variable with respect to class C of the Y variable.

When using linguistic variables, we must define the calculation of this value. In [12], a general methodology of belief calculation using fuzzy information was proposed. With this formulation, the probability of X taking a value a_i on its domain $\{a_1, a_2, \ldots, a_s\}$ is defined as

$$p(X = a_i) = \frac{1}{m} \sum_{j=1}^{m} \left(\frac{\mu_{a_i}(e_j)}{\sum_t \mu_{a_t}(e_j)} \right)$$

where

m	number of examples from the training set E;
e_j	example from E;
μ_w	membership function to the fuzzy set w.

In this formula, we assume that all the examples are crisp. The bidimensional probability is similar to the previous formula but requires the information on two variables to be combined using a t-norm (denoted by the symbol $*$)

$$p(X = a_i, Y = b_j) = \frac{1}{m} \sum_{k=1}^{m} \left(\frac{\mu_{a_i}(e_k) * \mu_{b_j}(e_k)}{\sum_{t, h} \mu_{a_t}(e_k) * \mu_{b_h}(e_k)} \right)$$

where the domain of variable Y is $\{b_1, b_2, \ldots, b_r\}$. From now on, we will use the minimum t-norm ($a * b = \min\{a, b\}$) in the previous formula.

Thus, we can use $\tau_C(X_i)$ as a measure of the initial relevance of variable X_i for the class C. This relevance measure will be included in the initial code for the variable chromosome on each individual of the population instead of a 0 or 1 default value. In this new approach, the value chromosome keeps the same description. When compared with the previous approach, the main difference is the code for the variable chromosome. In the previous section, the code of this substructure of an element of the population consisted of a binary array, where a value 1 in the variable X_i implies the activation of the variable and 0 the nonactivation of the variable. In this section, we use a variable chromosome with a real value containing $\tau_C(X_i)$, which has a value between 0 and 1. This value allows us to make smoother changes between relevance and irrelevance, i.e., during the execution, the evolutionary process must define the tendency toward the relevance or irrelevance of the variables. The problem lies in the interpretation of this new real code. For example, what does $\tau_C(X_i) = 0.7$ mean?

B. Activation Threshold

A possible interpretation is obtained if we include an activation threshold inside the variable chromosome. The inclusion of relevance measures and activation thresholds has previously been used in feature selection processes, such as [23], but in our case, a different activation threshold will be assigned to each chromosome and learned during the evolution process.

Thus, a variable X_i will be considered to be a component of the antecedent of the rule for a class if $\tau_C(X_i) \geq T_j$, where T_j represents the activation threshold for the different X_i variables of the j individual of the population. Otherwise, the variable will be considered to be irrelevant for the antecedent.

Example 3: Using the same variables as in the previous example, the code

$$\mathtt{VAR}\ (0.5,\ 0.7,\ 0.1)\ \mathtt{T}_{threshold}\ 0.6\ \mathtt{VAL}\ ((001)(11000)(10))$$

represents the following antecedent:

$$X_2 \text{ is } \{A_{21},\ A_{22}\}.$$

A different threshold is considered in each chromosome. Initially, the activation threshold is randomly defined for each element of the population. T_j takes a value in the interval

$$[\min_i \tau_C(X_i),\ \max_i \tau_C(X_i)].$$

The values T_j are affected by the genetic operators during the evolution of the GA and therefore the activation thresholds are learned by the algorithm.

The composition of the genetic operators will be introduced in the next section.

C. Genetic Operators

The set of operators for the value chromosome is exactly the same as the one we described in the previous section (genetic operators based on binary coding). The operators that modify the variable chromosome and the threshold (genetic operators based on real coding) are: the nonuniform mutation [30] and the BLX_α crossover operator [10].

- **Nonuniform mutation:** Let us suppose that $C = (c_1,\ \ldots,\ c_i,\ \ldots,\ c_s)$ is a chromosome. The nonuniform mutation alters a gene c_i in the following way:

$$c_i' = \begin{cases} c_i + \Delta(t,\ b_i - c_i), & \text{if } \beta = 0 \\ c_i - \Delta(t,\ b_i - a_i), & \text{if } \beta = 1 \end{cases}$$

where β is a binary random value, $[a_i,\ b_i]$ is the set of values that can be taken by the gene c_i, and

$$\Delta(t,\ y) = y\left(1 - r^{(1-(t/L))^b}\right)$$

where

r	random number in [0,1];
L	maximum number of iterations;
t	current iteration;
b	parameter selected by the user that determines the dependency degree with the number of iterations.

- BLX_α **crossover:** Let us suppose that $C_1 = (c_1^1,\ c_2^1,\ \ldots,\ c_s^1)$ and $C_2 = (c_1^2,\ c_2^2,\ \ldots,\ c_s^2)$ are two chromosomes selected for the crossover process. The BLX_α operator generates a new chromosome, $H = (h_1,\ \ldots,\ h_s)$ where each h_i is a number randomly selected in the interval $[c_{\min} - I\alpha,\ c_{\max} + I\alpha]$, where $c_{\max} = \max(c_i^1,\ c_i^2)$, $c_{\min} = \min(c_i^1,\ c_i^2)$, $I = c_{\max} - c_{\min}$ and α is a user-defined parameter to determine the spreading of the previous interval.

Finally, the value of τ_C is recalculated whenever the algorithm obtains a rule since when this happens, due to the architecture of SLAVE, the examples covered by the rules are eliminated and the training set is changed.

V. Experimental Studies

In this section, we consider the performance of the new proposals for learning algorithms. We have carried out empirical studies on this different proposal where we have the following:

- SLAVE is the original system without feature selection criterion.
- 2SLAVE-1 is the modified system described in Section III (binary code).
- 2SLAVE-2 is the modified system described in Section IV-B (activation threshold).

The following databases have been obtained from the UCI repository of machine learning databases and domain theories [31]:

- *The IONOSPHERE Data*: This radar data was collected by a system in Goose Bay, Labrador. The system consists of a phased array of 16 high frequency antennas with a total transmitted power to the order of 6.4 kW. The targets were free electrons in the ionosphere. There are two classes and 34 continuous attributes plus the class variable, using 351 examples.
- *SOYBEAN Database*: These data correspond to the information used to develop an expert system for soybean disease diagnosis. There are 19 classes and 35 categorical attributes, some nominal and some ordinal. The number of instances is 307 and there are missing values.[1]
- *WINE Recognition Data*: These data are the results of a chemical analysis of wines from the same region but with different types of grapes, using 13 continuous variables and 178 examples. This database contains three classes.
- *SONAR Database*: The task is to train a network to discriminate between sonar signals bounced off a metal cylinder and those bounced off a roughly cylindrical rock. There are two classes, 60 continuous inputs, and one enumerated output and 208 examples.
- *DERMATOLOGY Database*: The problem is the differential diagnosis of erythemato-squamous diseases. This database contains six classes, 34 attributes, 33 of which have linear values, one of them is nominal, and there are 366 examples.
- *PIMA Database*: The problem is the Pima Indians diabetes classification. This database contains two classes, eight attributes, and 768 examples.

Furthermore, we have used fuzzy domains on the different variables of each database, with each fuzzy domain being composed of seven fuzzy labels, uniformly distributed on its definition range.

We have run SLAVE and the different versions of 2SLAVE using the parameters described in Table I for each database.

For the different databases, we have used five training-test partitions (70% and 30%, respectively) obtained from the original database. For each training-test partition, we have calculated the following values:

- A: the accuracy (correct classification result on test sets);
- B: the number of rules;
- C: the average of variables in the different rules;
- D: the percentage of variables unused in none of the final sets of rules with respect to the total number of variables.

The above parameters are calculated using the arithmetic mean of the five test results. Moreover, since SLAVE is not a deterministic algorithm, the results of Table II correspond to the average of five different executions of each algorithm and on each training-test partition. The parameters of Table II therefore correspond to averages and standard deviations (the number after the $\pm$ symbol indicates the standard deviation of the reported average) on 25 different executions (five different training-test partitions and five different executions on its partition).

[1]SLAVE has been designed to work with missing values. Information about this feature can be found in [15].

TABLE I
SLAVE'S PARAMETERS

Parameter	Value
maximum number of iterations	500
number of iterations allowed without change	100
population size	20
mutation probability	0.01
crossover probability	0.6
AND operator	0.1
OR operator	0.1
Rotation operator	0.05
BLX_α crossover probability	0.6
Value of α parameter	0.15
k_1	0
k_2	1

Inspecting the results of the different versions of SLAVE tested in this experimental, we can see that the 2SLAVE-2 version obtained the best results since it presents the highest results in the four parameters studied.

- With respect to the parameter A (the correct classification result), 2SLAVE-2 clearly improves upon the accuracy of SLAVE and slightly improves upon the accuracy of 2SLAVE-1. This fact shows that overall the feature selection proposed in this work (2SLAVE-1 and 2SLAVE-2) actually improves accuracy. Moreover, the inclusion of information in the initial population (2SLAVE-2) allows us to obtain the same or better accuracy results.
- The parameter B (number of rules) is clearly reduced in the new versions when compared to the previous version. Although this number in 2SLAVE-2 is smaller than in 2SLAVE-1, there are no important differences between both versions.
- The analysis of the parameter C (average of used variables in the rules) is similar to the previous parameter. With the exception of the WINE database, the parameter is bigger in 2SLAVE-1 than in 2SLAVE-2 but the differences are not really important. The conclusion is that the new versions use a smaller number of variables in each rule than the original version of SLAVE.
- The last parameter D (proportion of variables unused in the final set of rules with respect to the total number of variables) is perhaps the most important one in connection with the proposal of this paper since it truly measures the feature selection capability of the new system. In this case, 2SLAVE-2 is clearly better than the other systems. The differences are very significant. For example, in the WINE database, the original systems do not use 5.32% of the variables, 2SLAVE-1 does not use 22.46% of the variables, but 2SLAVE-2 does not use 77.54% of the variables. This means that only 22.46% of the variables were used by some rule of the final set of rules. The average unused number of variables in the different databases is 55%.

From this discussion, we can deduce that the inclusion of the feature selection proposed in this paper and the inclusion of information about the initial relevance measure of the different variables in relation to the class, clearly produces very good results since it does not make the accuracy results worse (in fact, they are improved), it reduces the number of variables used in all the rules (eliminating irrelevant variables), and simplifies the description of the final set of rules.

We have also experimented with a simplified version of 2SLAVE-2, in which a random initialization has replaced to the information-based initialization, but we have obtained worse results.

Furthermore, we have used three well-known learning algorithms (C4.5, CN2, and LVQ) to compare the accuracy results. These algorithms represent three different learning methodologies.

TABLE II
EXPERIMENTAL RESULTS OF THE DIFFERENT VERSIONS OF SLAVE WHERE A IS THE ACCURACY, B THE NUMBER OF RULES, C THE AVERAGE OF VARIABLES IN THE DIFFERENT RULES, AND THE PERCENTAGE OF VARIABLES UNUSED IN NONE OF THE FINAL SETS OF RULES WITH RESPECT TO THE TOTAL NUMBER OF VARIABLES

		IONOSPHERE	SOYBEAN	WINE	SONAR	DERMATOLOGY	PIMA
SLAVE	A	72.81±3.94	94.42±0.16	87.40±3.93	48.72±2.56	83.30±3.28	76.46±0.99
	B	67.2 ±9.67	37.8 ±7.03	12.3 ±2.38	93.6 ±0.43	53.4 ±4.88	20.0 ±4.15
	C	30.42±0.04	32.42±0.07	12.13±0.30	49.9 ±0.01	13.96±0.03	4.33 ±0.12
	D	2.94±0.01	5.91 ±0.02	5.32 ±1.12	15.01±5.29	9.20 ±5.02	0.00 ±0.00
2SLAVE-1	A	90.58±2.92	98.73±0.96	89.17±5.20	65.86±4.55	92.99±1.77	77.60±0.95
	B	14.2 ±1.52	33.2 ±3.71	8.7 ±1.38	12.4 ±1.85	13.0 ±1.41	9.8 ±1.49
	C	2.69 ±0.32	2.41 ±0.13	2.24 ±0.21	3.77 ±0.49	2.56 ±0.29	2.98 ±0.42
	D	30.47±7.97	14.28±5.72	22.46±9.33	44.33±7.49	41.76±7.05	7.50 ±5.00
2SLAVE-2	A	90.91±2.34	98.42±1.75	90.48±3.76	74.83±3.55	94.43±1.05	78.12±1.14
	B	10.1 ±3.44	29.6 ±2.15	8.0 ±1.37	11.6 ±1.62	11.2 ±0.75	6.4 ±2.18
	C	2.46 ±0.63	2.35 ±0.16	2.30 ±0.32	2.67 ±0.22	2.39 ±0.07	2.05 ±0.37
	D	49.53±2.73	74.28±6.26	77.54±10.41	62.00±4.26	47.06±2.63	20.00±7.79

TABLE III
EXPERIMENTAL RESULTS OF OTHER LEARNING ALGORITHMS AND 2SLAVE-2

		IONOSPHERE	SOYBEAN	WINE	SONAR	DERMATOLOGY	PIMA
C4.5	accuracy	86.50	97.90	93.20	70.70	95.90	67.70
	n.nodes	37	65	10	27	15	163
CN2	accuracy	83.96	98.6	89.76	70.70	94.80	74.50
	n.rules	16	32	9	21	14	38
LVQ	accuracy	87.97	91.75	72.60	79.31	86.60	67.71
2SLAVE-2	accuracy	90.91	98.42	90.48	74.83	94.43	78.12
	average n.rules	10.1	29.6	8.0	11.6	11.2	6.4

- **C4.5:** This is an implementation of the well-known C4.5 classification algorithm based on classification trees and it is described in [32].
- **CN2:** This algorithm inductively learns a set of propositional rules from an example set. The algorithm, which was proposed in , is based on the methodology of the learning algorithm of the AQ family, and attempts to improve the behavior when the example set is affected by noise. The implementation used in this work was developed by Boswell in 1990 and can be obtained from http://www.cs.utexas.edu/users/pclark/software.html.
- **LVQ:** This is an adaptive learning method based on Kohonen self-organizing maps [24]. The implementation used in this work is version 3.1 of the LVQ-PAK, available at ftp://cochlea.hut.fi/pub/lvq-pak. This software contains all the necessary programs for the application of the algorithm learning vector quantization (LVQ) in statistical classification or pattern recognition. Among the different programs included in the distribution we have used LVQ1 software without the initialization process because it has shown the best behavior on the databases used. Furthermore, we have selected a third of the number of examples in the training set as the number of codevectors.

Table III shows the accuracy obtained when these learning algorithms are used on the five test sets.

Comparison of these results shows how the accuracy obtained by the different versions of SLAVE is competitive with the accuracy obtained by other well-known learning algorithms. In this case, the different versions of SLAVE obtain the best accuracy in three databases (IONOSPHERE, SOYBEAN, and PIMA), and in the other three databases, the results are not very different in terms of the best result.

VI. CONCLUDING REMARKS

We have proposed a modification of the GA of SLAVE that allows us to select the appropriate features for a problem. This modification dynamically explores the set of possible variables in order to find the most useful rule and the most interesting variables for this rule.

The basic schema consists in modifying the representation of a rule in the search mechanism of SLAVE in such a way that in the new version, the learning algorithm can search for not only the best rule but also the best set of variables for each rule. The experimentation with the databases considered has shown that the feature selection of the new genetic learning algorithm has clearly improved. Moreover, the new learning algorithms (2SLAVE-1 and 2SLAVE-2) obtain better accuracy results and simpler rules. The experimentation has also shown how the inclusion of information about the initial relevance of the variables (2SLAVE-2) obtains a clear improvement in relation to the feature selection.

It is important to point out that the feature selection studied in this paper has been carried out on a rule-to-rule basis unlike the classical methods that obtain a set of variables for all the rules. Thus, if we were to include in a set all the variables that appear in any rule, this set would contain all the used variables, as in other feature selection models. The proposed model is however more general since it can obtain a set of rules but moreover, only the useful information for each class is used in each particular rule.

Acknowledgment

The authors would like to thank the anonymous referees for their useful corrections and suggestions.

References

[1] J. E. Baker, "Adaptive selection methods for genetic algorithm," in *Proc. First Int. Conf. Genetic Algorithms Applicat.*, J. J. Grefenstette, Ed. Hillsdale, MA: Lawrence Erlbaum, 1985, pp. 101–111.

[2] P. Clark and T. Niblett, "The CN2 induction algorithm," *Mach. Learn.*, vol. 3, no. 4, pp. 261–283, 1989.

[3] E. I. Chang and R. P. Lippmann, "Using genetic algorithms to improve pattern classification performance," *Neural Inform. Process. Syst.*, vol. 3, pp. 797–803, 1991.

[4] O. Cordón and F. Herrera, "A three-stage evolutionary process for learning descriptive and approximate fuzzy-logic-controller knowledge bases from examples," *Int. J. Approx. Reasoning*, vol. 17, pp. 369–407, 1997.

[5] O. Cordón *et al.*, "Genetic learning of fuzzy rule-based classification systems cooperating with fuzzy reasoning methods," *Int. J. Intell. Syst.*, vol. 13, pp. 1025–1053, 1998.

[6] O. Cordón and F. Herrera, "A two-stage evolutionary process for designing TSK fuzzy rule-based systems," *IEEE Trans. Syst., Man, Cybern. B*, vol. 29, pp. 703–715, Dec. 1999.

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

[8] K. A. De Jong and W. M. Spears, "A formal analysis of the role of multipoint crossover in genetic algorithm," *Ann. Math. Artif. Intell.*, vol. 5, no. 1, pp. 1–26, 1992.

[9] K. A. De Jong *et al.*, "Using genetic algorithms for concept learning," *Mach. Learn.*, vol. 13, pp. 161–188, 1993.

[10] L. J. Eshelman and J. D. Schaffer, "Real coded genetic algorithm and interval schemata," in *Foundations of Genetic Algorithm*, L. D. Whitley, Ed. San Mateo, CA: Morgan Kaufmann, 1993, vol. 2, pp. 187–202.

[11] T. Furuhashi *et al.*, "A study on fuzzy classifier system for finding control knowledge of multi-input systems," in *Genetic Algorithms and Soft Computing*, F. Herrera and J. L. Verdegay, Eds. New York: Physica-Verlag, 1996, pp. 489–504.

[12] A. González, "A learning methodology in uncertain and imprecise environments," *Int. J. Intell. Syst.*, vol. 19, pp. 357–371, 1995.

[13] A. González and F. Herrera, "Multi-stage genetic fuzzy systems based on the iterative rule learning approach," *Mathw. Soft Comput.*, vol. 4, no. 3, pp. 233–249, 1997.

[14] A. González and R. Pérez, "A learning system of fuzzy control rules," in *Genetic Algorithms and Soft Computing*, F. Herrera and J. L. Verdegay, Eds. New York: Physica-Verlag, 1996, pp. 202–225.

[15] ——, "Completeness and consistency conditions for learning fuzzy rules," *Fuzzy Sets Syst.*, vol. 96, no. 1, pp. 37–51, 1998.

[16] ——, "SLAVE: A genetic learning system based on an iterative approach," *IEEE Trans. Fuzzy Syst.*, vol. 27, pp. 176–191, Apr. 1999.

[17] A. González *et al.*, "Learning the structure of a fuzzy rule: A genetic approach," in *Proc. EUFIT*, vol. 2, Aachen, Germany, 1993, pp. 814–819.

[18] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: Univ. of Michigan Press, 1975.

[19] J. H. Holland and J. S. Reitman, "Cognitive systems based on adaptive algorithms," in *Pattern-Directed Inference Systems*, D. A. Waterman and F. Hayes-Roth, Eds. New York: Academic, 1978.

[20] H. Ishibuchi and T. Murata, "A genetic-algorithm-based fuzzy partition method for pattern classification problems," in *Genetic Algorithms and Soft Computing*, F. Herrera and J. L. Verdegay, Eds. New York: Physica-Verlag, 1996, pp. 555–578.

[21] C. Z. Janikow, "A knowledge-intensive genetic algorithm for supervised learning," *Mach. Learn.*, vol. 13, pp. 189–228, 1993.

[22] J. Jelonek and J. Stefanowski, "Feature subset selection for a classification of histological images," *Artif. Intell. Med.*, vol. 9, pp. 227–239, 1997.

[23] K. Kira and L. A. Rendell, "The feature selection problem: Traditional methods and a new algorithm," in *Proc. Ninth National Conf. Artificial Intell.*, vol. 1, San Jose, CA, 1992, pp. 129–134.

[24] T. Kohonen, *Self-Organizing Maps*, Second Extended ed, ser. Springer Series in Information Sciences. New York: Springer-Verlag, 1995, 1997, vol. 30.

[25] S. Kullback, *Information Theory and Statistics*. New York: Dover, 1968.

[26] A. L. Blum and P. Langley, "Selection of relevant features and examples in machine learning," *Artif. Intell.*, vol. 97, pp. 245–271, 1997.

[27] R. Leardi *et al.*, "Genetic algorithms as a strategy for feature selection," *J. Chemometrics*, vol. 6, pp. 267–281, 1992.

[28] L. Magdalena and F. Monasterio, "A fuzzy logic controller with learning through the evolution of its knowledge base," *Int. J. Approx. Reasoning*, vol. 16, pp. 335–358, 1997.

[29] L. Magdalena, "Adapting the gain of an FLC with genetic algorithms," *Int. J. Approx. Reasoning*, vol. 17, pp. 327–349, 1997.

[30] Z. Michalewicz, *Genetic Algorithms $+$ Data Structures $=$ Evolutive Programs*. New York: Springer-Verlag, 1992.

[31] P. M. Murphy and D. W. Aha, "UCI repository of machine learning databases," Univ. California, Dept. Info. Comput. Sci., Irvine, CA, http://www.ics.uci.edu/~mlearn/MLRepository.htm, 1998.

[32] J. R. Quinlan, *C4.5 Program for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1993.

[33] W. Siedlecki and J. Sklansky, "A note on genetic algorithms for large-scale feature selection," *Pattern Recognit. Lett.*, vol. 10, pp. 335–347, 1989.

[34] S. F. Smith, "A learning system based on genetic adaptive algorithms," Ph.D. dissertation, Univ. of Pittsburgh, Dept. Comput. Sci., Pittsburgh, PA, 1980.

[35] H. Vafaie and K. A. De Jong, "Genetic algorithms as a tool for feature selection in machine learning," in *Proc. Int. Conf. Tools AI*, Arlington, VA, 1992, pp. 200–204.

[36] G. Venturini, "SIA: A supervised inductive algorithm with genetic search for learning attributes based concepts," *Mach. Learn.*, pp. 280–296, 1993.

[37] L. Wehenkel, "On uncertainty measures used for decision tree induction," in *Proc. IPMU*, vol. 1, Granada, Spain, 1996, pp. 413–418.

[38] L. A. Zadeh, "The concept of a linguistic variable and its applications to approximate reasoning," *Inform. Sci.*, vol. 8, pp. 199–249.