Evolving Complex Fuzzy Classifier Rules Using a Linear Tree Genetic Representation

Dipankar Dasgupta

Division of Computer Science
Mathematical Sciences Department
The University of Memphis
Memphis, TN 38152.
ddasgupt@memphis.edu

Fabio A. González

Division of Computer Science
Mathematical Sciences Department
The University of Memphis
and Universidad Nacional de Colombia
fgonzalz@memphis.edu

Abstract

The paper proposes a linear representation of tree structures in order to evolve complex fuzzy rule sets for solving classification problems. In particular, linguistic rules are evolved, where the condition part of a rule can have an arbitrary structure of conjunctions and disjunctions. We describe an efficient rule representation scheme, which uses a genetic algorithm. The method is tested with a number of benchmark data sets and some results are reported.

1 INTRODUCTION

The problem of classification has been studied extensively in machine learning (Holte,93; Michalski,98; Weiss,91), and has recently received a lot of attention in the emerging area of Data Mining (or Knowledge Discovering) (Han,00; Michalski,98). The problem of classification can be stated as follows: Given a set of classified elements (training set), build a system (classifier) that is capable of categorizing unlabeled elements (testing set) where the label of an element represents the class to which it belongs.

There exist many approaches for solving classification problems (Weiss,91): statistical methods, decision trees, neural networks, rule-based methods, etc.; and all of them have some advantages and disadvantages (Curram,94; Lim,97). The choice of a particular method depends, however, on factors like the kind of problem to be solved, the resources available, etc. An important factor in a good number of problems is the comprehensibility of the resulting classifier, that is, the possibility of understanding the resulting model and extracting useful knowledge to understand the modeled system. Approaches like neural networks and many of the statistical methods have very little comprehensibility (Weiss,91).

Another method, Fuzzy logic has been applied successfully (Fidelis,00; Gonzalez,98; Ishibuchi,00, 97 and 95) to extract comprehensible classifier knowledge from data in the form of linguistic rules (in this context, the linguistic is synonym of fuzzy). The fuzzy method has the ability to represent imprecise knowledge and the capability of dealing with noisy data.

Moreover, there have been several works that have attempted to produce classifier rules (fuzzy and non-fuzzy) using evolutionary techniques (Bojarczuk,99; Ishibuchi,95; Liu,00). One of the main problems encountered in this approach, is the representation of the condition part of a rule in the chromosome. Since the condition part can be a very complex logical expression, there is not a natural way to represent it as linear string. However, there are two main approaches that have been studied (De Jong,91):

- *Linear representation of the condition part*

  In some approaches (De Jong,91; Fidelis,00; Gonzalez,1998; Ishibuchi 00 and 95; Liu,00), the condition part was restricted to be a conjunction of one or more logical terms (tests). This makes the representation of the condition as a linear string. But, in general, a single rule is not sufficient enough to characterize a class; rather a set of rules is necessary. In other approaches (Giordana, 93), condition structures were predefined and only some parameters of rules were evolved. Figure 1(a) shows an example of such cases.

- *Tree representation of the condition part*

  In this approach, it was possible to represent arbitrarily complex conditions using Genetic Programming, with a substantial increase in the implementation complexity (Bojarczuk,99; Folino,99; Freitas,97; Tunstel,96). Figure 1(b) gives an example of tree representation.

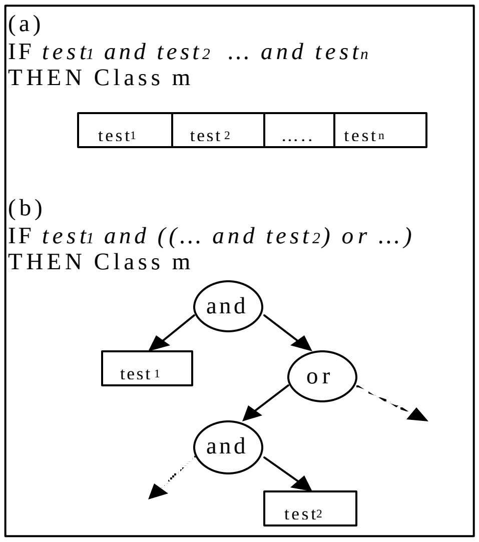
(a)
IF test1 and test2 ... and testn
THEN Class m
test1
test2
.....
testn
(b)
IF test1 and ((... and test2) or ...)
THEN Class m
and
test1
or
and
test2

Figure 1: Conventional approaches to represent condition part of a rule. (a) a linear representation (b) a tree representation.

The purpose of the work presented in this paper, is to explore a new representation for linguistic classifier rules, which tries to combine the linear and tree methods, exploiting the advantages of both. In this approach, we evolve arbitrarily complex rules using a novel representation of tree structures in order to apply a genetic search.

The subsequent sections are organized as follows. Section 2 briefly describes the approach to perform classification tasks using fuzzy IF-THEN rules. Section 3 presents the proposed fuzzy rule representation scheme, Section 4 describes experiments and the analysis of results, and Section 5 draws some conclusions.

2 CLASSIFICATION USING LINGUISTIC RULES

In general, a linguistic classifier rule has the following form:

IF x_1 Î S_1 op_1 x_2 Î S_2 ... op_{n-1} x_n Î S_n THEN *Class m*

where,

$\underline{x_i} \in [0.0,1.0]$, is an attribute or linguistic variable

$S_i \in \{S,MS,M,ML,L\}$, is a fuzzy set

$op_i \in \{AND, OR\}$, is a Fuzzy-Boolean operator

In this work, the attribute values are normalized in the interval [0.0,1.0] and fuzzy sets are defined by the membership functions shown in the Figure 2.

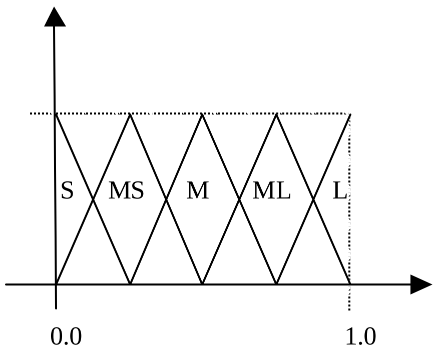

Figure 2: Fuzzy sets and membership functions

In our current experiments, we used 5 linguistic values, such as S (small), MS (medium small), M (medium), ML (medium large) and L (large). However, the method can be easily extended to any number of fuzzy values.

A classifier model can be represented by a set of *m* rules, where *m* is the number of different classes, that is, each class is represented by one, and only one, rule. For example,

R_1: IF $Condition_1$ THEN Class C_1

: : : :

R_m: IF $Condition_m$ THEN Class C_m

In order to classify an unclassified element $(x_1, \ldots, x_n)$, which is represented by a vector of attributes, the condition part of each rule is evaluated using the membership functions and the fuzzy-set operators[1]. Then, the rule with the highest value in the condition is selected, and the element is classified according to the consequent part of that rule:

$$Class(x_1,\ldots,x_n) = \max_{c\in\{1,\ldots,m\}} \{Condition_c(x_1,\ldots,x_n)\}$$

where, $Condition_m(x_1, \ldots, x_n)$ represents the value of the $Condition_C$ evaluated for the element $(x_1, \ldots, x_n)$, which is a real value between 0.0 and 1.0.

3 PROPOSED APPROACH

In general, the condition part of a rule corresponds to a logic expression, which can be represented by an expression tree; a linear chromosome with variable length represents this expression tree.

A standard genetic algorithm with special operators is applied to evolve the rules. A GA run evolves a rule, so multiple runs are needed to cover all classes in the training set. The elements in the training set that belong to

[1] The union (OR) operator is calculated by the function *max(,)* and the intersection (AND) by the function *min(,)*

the class of the respective run are considered positive examples and the elements that belong to other classes are considered negative examples.

3.1 LINEAR REPRESENTATION OF LINGUISTIC RULES

Since there are different GA runs for each class, we do not have to represent the action part of the rule in the chromosome; it only represents the condition part.

Formally, a condition is generated by the following grammar:

```
(1) <condition>    ::=  <condition> <operator>
                        <condition>
                   |    <atomic_condition>
(2) <atomic_cond>  ::=  <variable> <rel op>
                        <set>
(3) <operator>     ::=  AND <prec> | OR <prec>
(4) <variable>     ::=  X1|...| xn
(5) <rel op>       ::=  ∈ | ∉
(6) <set>          ::=  S | MS | M | ML | L
(6) <prec>         ::=  1 | 2 |...| 8
```

The tree structure of an expression is generally expressed using braces that indicate the order of evaluation of the operators. When braces are not used, the default precedence of the operators determines the order of evaluation.

In our approach, we introduced precedence values for each operator in the representation itself (represented by `<prec>` in the grammar). This precedence value indicates the order of evaluation; an operator with a higher precedence value is evaluated first. Therefore, it is not necessary to have braces or a tree representation to express the evaluation order, so the expression can be represented by a linear string.

For example, the condition

X_2 *Î MS AND*$_2$ X_1 *Ï S OR*$_1$ X_3 *Î ML AND*$_3$ X_2 *Î L*

represents the condition expression, as shown in Figure 3:

(X_2 *Î MS AND* X_1 *Ï S) OR (*X_3 *Î ML AND* X_2 *Î L)*

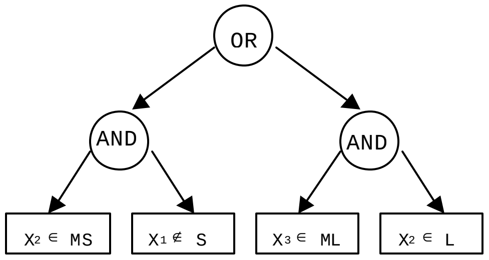

Figure 3: Tree representation of a condition expression

The precedence value of the operator *AND*$_2$ indicates that this operation has to be performed before the operation *OR*$_1$. When two consecutive operators have the same precedence value, the left one is evaluated first.

This scheme allows the representation of arbitrary complex conditions; the number of different precedence values determines the maximum depth of an expression tree.

Applying the grammar rule (1) multiple times, we get a condition with the following structure:

<ac$_1$> <op$_1$> ... <ac$_n$> <op$_n$> <ac$_{n+1}$>

where

<ac$_i$>: Atomic Condition

<op$_i$>: Fuzzy Operator

This condition expression is represented by a chromosome with the structure shown in the Figure 4.

Gene$_1$					...	Gene$_n$					Gene$_{n+1}$			
ac$_1$			op$_1$		...	ac$_n$			op$_n$		ac$_{n+1}$			**
var$_1$	ro$_1$	s$_1$	o$_1$	prec$_1$	...	var$_n$	ro$_n$	s$_n$	o$_n$	prec$_n$	var$_{n+1}$	ro$_{n+1}$	s$_{n+1}$	**

Figure 4: Chromosome representation of the condition.

An Atomic Condition and a Fuzzy Operator compose a gene. However, there is an exception in the last gene, which is composed of an Atomic Condition, and the last part (Fuzzy Operator) is ignored.

In our implementation, each gene is represented using 16 bits in the following way:

- Atomic Condition part:
  - 8 bits to represent the variable (var$_i$)
  - 1 bit to represent the relational operator (ro$_i$)
  - 3 bits to represent the set (s$_i$)
- Operator part:
  - 1 bit to distinguish between AND and OR (o$_i$)
  - 3 bits to represent the precedence (prec$_i$)

An important characteristic of this representation is that, in order to express the genotype, it is not necessary to build the expression tree. Instead, the classical parsing algorithm, *operator precedence parser* (Aho,86), can be used. This technique allows the evaluation of an expression in a very efficient way. The chromosome only has to be traversed once, that is, the time complexity of the evaluation is *O(n)*, where *n* is the condition expression length.

The evaluation algorithm based on the operator precedence parser can efficiently be implemented (using array and stack operations instead of pointer operations).

This fact along with the compact chromosomal representation makes this approach computationally inexpensive.

3.2 FITNESS EVALUATION

The fitness of each chromosome (rule) is evaluated with respect to a set of attribute vectors (training set) to which a class has been previously assigned. In each run of the genetic algorithm, a rule with different class C_i is evolved. Accordingly, vectors in the training set with class part equal to C_i are considered positive examples, and the elements with class part different from C_i are considered negative examples.

In our approach, the first step is to evaluate the condition part of the rule for a given vector. If the result is greater than or equal to 0.5, then the condition is true, otherwise it is false. Next, the class of the vector is compared to the class C_i of the actual run, and four different outcomes are possible, shown in Table 1.

Table 1: Types of the classifications results

Condition	Class	Type
TRUE	Equal	True Positive (TP)
TRUE	Different	False Positive (FP)
FALSE	Equal	False Negative (FN)
FALSE	Different	True Negative (TN)

The fitness of the condition is evaluated taking into account three objectives: maximize the sensitivity, maximize the specificity, and minimize the length of the chromosome. The length of the chromosome is penalized, because we want to evolve simple rules. This is an important factor that contributes to the comprehensibility. The formulas used are as follow:

$$sensitivity = \frac{TP}{TP + FN}$$

$$specifcity = \frac{TN}{TN + FP}, \text{ and}$$

$$fitness = w_1 \cdot sensitivity + w_2 \cdot specifcity + w_3 (1 - \frac{length}{MaxLength})$$

where *MaxLength* is the maximum allowable genes in a chromosome, and *length* is the actual number of genes in the chromosome.

This is a multi-objective problem, and there are different ways to deal with this kind of problem (Fonseca, 97). We chose to use a weighted sum approach, however, further experimentation with other multi-objective optimization approaches will be necessary. The w_i terms in the fitness definition represent the weight values.

3.3 GENETIC OPERATORS

The following genetic operators are used:

- **Restricted Crossover**: A crossover point is chosen between 1 and the minimum of the lengths of the two selected chromosomes. The child with minimal length is chosen (Figure 5.a).
- **Mutation**: A randomly chosen bit is changed as used in simple GA's.
- **Gene Elimination**: A gene is chosen randomly and eliminated. The length of the new chromosome is 16 bits shorter than the parent chromosome (Figure 5.b).
- **Gene Addition**: A random gene is generated and added at the end of the chromosome. The length of the new chromosome is 16 bits longer than the parent chromosome (Figure 5.c).

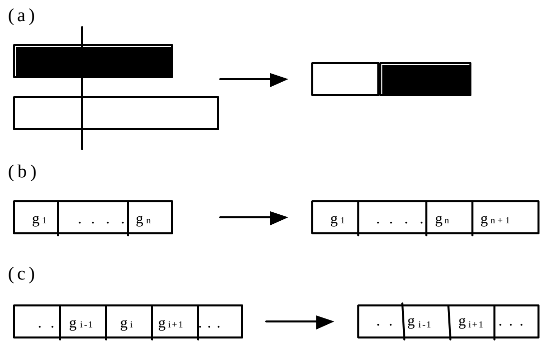
(a)
(b)
g_1 . . . . g_n
g_1 . . . . g_n g_{n+1}
(c)
. . g_{i-1} g_i g_{i+1} . . .
. . g_{i-1} g_{i+1} . . .

Figure 5: Genetic operators. (a) restricted crossover (b) gene addition (c) gene elimination.

However, only one operator is applied each time. The operator to be applied is chosen using a uniformly generated random number and the probability assigned to each operator.

4 EXPERIMENTATION

In order to evaluate the performance of the proposed approach to extract comprehensible linguistic rules from the training data, tests were conducted using publicly available data sets (University of California, Irvine, Repository of Machine Learning Databases (Blake,98)). These data sets are referenced frequently in the classification and machine learning literature, and it is a well-known standard for testing.

The data sets used are described in Table 2. The sample size s, the number of classes, and the type of attributes are shown in Table 2.

Table 2: Test Data sets used for experiments

Data Set	Sample size	No. of classes	Attributes	
			Numerical	Categorical
IRIS	150	3	4	0
VOTE	435	2	0	16
WINE	178	3	13	0

4.1 EXPERIMENTAL SETTING

The data sets with numerical attributes were normalized to have all values in a fixed range [0.0,1.0]. The attributes of the VOTE data set have 3 possible values 'YES', 'NO' and '?'[2]; these values were codified as 1.0, 0.0 and 0.5, respectively, to deal with categorical data.

A ten-fold testing strategy was employed (Lim, 97), that is, the data set was partitioned into ten randomly chosen subsets, and each subset was used as a testing set for the classifier trained with the remaining subsets. The score of the classifier (*correctly classified samples / sample size*) was calculated as the average score of 10 tests. This process was repeated 5 times for each data set and the average score was taken.

A number of GA parameters were tested, and the reported results used tournament selection, with a tournament size of 4, along with elitism -- the best individual of each generation is copied to the next generation.

GA parameter values:

Population:	200
Generations:	200
Mutation Rate:	0.05
Crossover Rate:	0.35
Gene Addition Rate:	0.35
Gene Elimination Rate:	0.25
Maximum Length:	50 genes

Each GA run was initialized with a random population of rules with five genes. The weights used in the fitness function were w_1=0.45, w_2=0.45 and w_3=0.1, to give more importance to sensitivity and specificity terms. In our empirical study, these values produced good results in different experiments; however, more experimentation will be necessary to define criteria for tuning parameter values.

4.2 RESULTS AND ANALYSIS

The average score and the variance in data sets are reported in Table 3. In particular, the value 94%+/-0.3 in the first row illustrates that the experimentations of the IRIS data set produce an average score of 94.5% with a variance of 0.3%. Although quantitative comparisons with other methods are useful, and desirable, our results compare well to those reported in the literature (IRIS (Folino,99; Gonzalez,1998; Ishibuchi,95; Liu,00), VOTE (Folino,99; Lim,97), WINE (Ishibuchi,00)).

Table 3: Results of average prediction accuracy

Data Set	Score
IRIS	94.5% +/- 0.3
VOTE	94.7% +/- 0.1
WINE	93.9% +/- 0.7

The most important objective of our fuzzy rule classification was to obtain comprehensible rules. The proposed approach was able to evolve simple rules, and the following set of rules were evolved for IRIS data set in a typical run:

R1: if X_3 Î *S* OR_6 X_2 Î *S* THEN Class 1

R2: if X_3 Î *M* THEN Class 2

R3: if X_3 Î *ML OR (* X_2 Î *ML AND* X_0 Î *M) OR* X_3 Î *L* THEN Class 3

There are two key points to be noted in this set of rules:

- The simplicity of the rules: The system was able to evolve short rules.
- The feature selection: The system was able to select attributes that contribute to distinguishing a class, discarding other attributes.

These improved results were accomplished due to the evolutionary pressure towards shorter rules. Figure 6 shows the evolution of the fitness and the average rule size of the population in a particular run of the GA using the IRIS data set. It shows how the size of the rule converges to an optimal value, while the average fitness of the population increases.

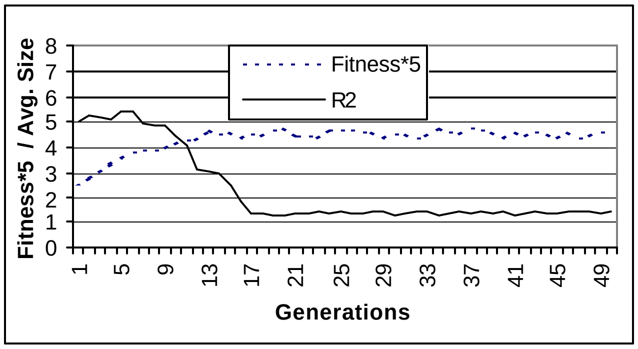

Figure 6: Fitness and size of the rule 2 evolution for the IRIS data set (the fitness is multiplied by 5)

[2] The character '?' means a neutral vote, neither YES or NO.

The optimal size of a rule depends on the complexity of the class that it intends to model. Figure 7 shows the change in the average size of three rules for the IRIS data set. The convergence value of each rule is different and it can be interpreted as a measure of complexity of the class that it represents.

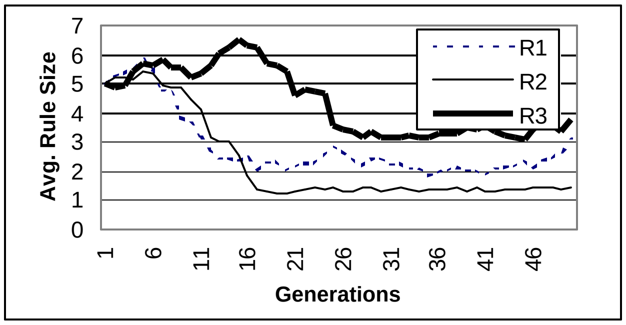

Figure 7:Evolution of the rule size for the IRIS data set.

Figure 8 presents the rule size behavior for the WINE data set. The convergence values are greater than those of the IRIS data set. This indicates that the WINE data set is more complex than the IRIS data set from the classification point of view.

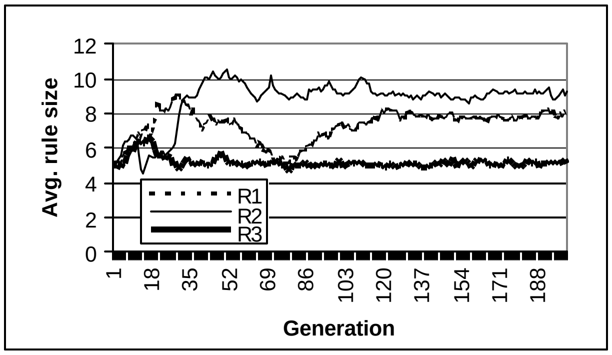

Figure 8: Evolution of the rule size for the WINE data set.

In some cases, there is a tradeoff between simplicity and accuracy. Table 4 shows the scores for the previously shown set of evolved rules for the IRIS data set. Here, the rule R2 has 4 false positives and 2 false negatives. In another run with the same training set (IRIS), a different rule was evolved for the class 2:

R2′:IF (X_2 Î M OR X_1 Ï S) AND X_3 Î M

THEN Class 2

Table 4: Fitness of evolved rules for the IRIS data set. Here ***Sens, Spec*** and ***Fitn*** represent sensitivity, specificity and fitness, respectively.

Rule	**TP**	**TN**	**FP**	**FN**	**Sens**	**Spec**	**Fitn**
R1	42	93	0	0	100%	100%	0.996
R2	44	85	4	2	96%	96%	0.958
R3	47	80	8	0	100%	91%	0.951

This rule (R2′) has a more restrictive condition than previous R2. In effect, the number of false positives is reduced to 3, but the fitness remains similar (0.959) because of the increase of the condition length (chromosome).

The simplicity of the rules depends on the data set characteristics. For instance, the rules evolved for the WINE data set are more complex. The following is a typical rule evolved for this data set:

IF

(X_3 Î MS OR X_{10} Î M) AND (X_0 Ï MS OR X_{10} Î S) AND X_5 Ï MS AND X_9 Ï S AND X_6 Ï MS AND X_4 Ï S

THEN Class 1

In the case of the VOTE data set, there are only two classes. The complexity of these two rules is expected to be the same, and the experiments confirmed that as is shown in figure 9.

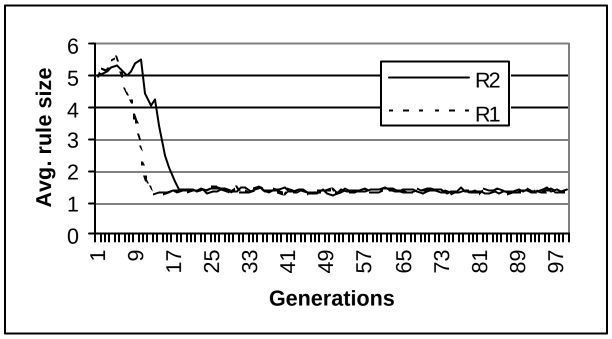

Figure 9: Average rule size for the VOTE data set.

5 CONCLUSIONS

Our experiments showed that the proposed representation works well in a wide variety of classification problems. Despite the fact that only five values for the linguistic variables were used, the accuracy of the evolved classifier rules was very good and comparable to those reported in the literature. The accuracy can be further improved by increasing the number of linguistic values and applying genetic tuning methods to the membership functions (Herrera, 98).

The main goal of this work was to evolve comprehensible rules, which could be accomplished by producing shorter rules, and performing automatic feature selection according to the complexity of data.

The main contribution of the present work is the design of a representation scheme. It allows an efficient and compact representation of complex conditions, using a linear chromosome.

However, more experiments need to be performed with bigger data sets and using other genetic operators. It is also important to perform quantitative comparison against other rule evolution methods, which is a part of our future work.

Acknowledgements

This work was funded by the Defense Advanced Research Projects Agency (no. F30602-00-2-0514) and Office of Navel Research (no. N000149910721).

References

A.V. Aho, R. Sethi, and J.D. Ullman (1986). *Compilers: Principles, Techniques, and Tools.* Addison-Wesley.

C.L. Blake, and Merz, C.J. (1998). UCI Repository of machine learning databases Irvine, CA: University of California, Department of Information and Computer Science. [http://www.ics.uci.edu/~mlearn/MLRepository.html].

C.E. Bojarczuk, H.S. Lopes and A.A. Freitas (1999). Discovering comprehensible classification rules using genetic programming: a case study in a medical domain. *Proc. Genetic and Evolutionary Computation Conference GECCO99*, Morgan Kaufmann, 1999, pp. 953-958.

S. Curram and J. Mingers (1994). Neural Networks, Decision Tree Induction and Discriminant Analysis: An empirical comparison. *J. of the Operational Research Society*, **45**(4):440-450.

K. De Jong and W. Spears (1991). Learning Concept Classification Rules Using Genetic Algorithms. *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pp. 651-656.

M.V. Fidelis, H.S. Lopes and A.A. Freitas (2000). Discovering comprehensible classification rules with a genetic algorithm. *Proc. Congress on Evolutionary Computation (CEC)*, pp. 805-810.

G. Folino, C. Pizzuti and G. Spezzano (1999). A Cellular Genetic Programming Approach to Classication. *Proc. Of the Genetic and Evolutionary Computation Conference GECCO99*, Morgan Kaufmann, pp. 1015-1020.

C.M. Fonseca and P.J. Fleming (1997). Multiobjective Optimization. In *Handbook of Evolutionary Computation*, release 97/1, IOP Publishing Ltd. and Oxford University Press.

A.A. Freitas (1997). A Genetic Programming Framework for two Data Mining Tasks: Classification and Generalised Rule Induction. *Proc. of 2nd Annual Genetic Programming Conference GP'97*, pp 96-101.

A. Giordana and L. Saitta (1993). Regal: an integrated system for learning relations using genetic algorithms. *Proceedings of the Second International Workshop on Multistrategy Learning*. R.S. Michalski et G. Tecuci (eds), pp. 234-249.

A. Gonzalez and R. Prez (1998). Completeness and consistency conditions for learning fuzzy rules. *Fuzzy Sets and Systems*, 96: 37-51.

J. Han and M. Kamber (2000). *Data Mining: Concepts and Techniques*. Morgan Kaufmann.

F. Herrera, M. Lozano and J.L. Verdegay (1998). A learning process for fuzzy control rules using genetic algorithms. *Fuzzy Sets and Systems*, 100:143-158.

R. Holte (1993). Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63-91.

H. Ishibuchi and T. Nakashima (2000). Linguistic Rule Extraction by Genetics-Based Machine Learning. *Proceedings of the Genetic and Evolutionary Computation Conference GECCO'00*, 195-202. Morgan Kaufmann.

H. Ishibuchi and T. Murata (1997). A genetic-algorithm-based fuzzy partition method for pattern classification problems. In F. Herrera and J.L. Verdegay (eds.), *Genetic algorithms and soft computing*, Physica-Verlag, pp. 555-578.

H. Ishibuchi, K. Nozaki, N. Yamamoto and H. Tanaka (1995). Selecting fuzzy if-then rules for classification problems using genetic algorithms. *IEEE Transactions on Fuzzy Systems*, **3**(3):260-270.

T. Lim and W. Loh (1997). *A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-Three Old and New Classification Algorithms*. Technical Report, Department of Statistics, University of Wisconsin-Madison, No. 979.

J. Liu and J. Kwok (2000). An extended genetic rule induction algorithm. *Proceedings of the Congress on Evolutionary Computation (CEC)*, pp.458-463.

R.S. Michalski, I. Bratko and M. Kubat (1998). *Machine learning and data mining: methods and applications*. J. Wiley & Sons, U.K.

E. Tunstel and M. Jamshidi (1996). On Genetic Programming of Fuzzy Rule-Based Systems for Intelligent Control. *Intl. Journal of Intelligent Automation and Soft Computing*, **2**(3):271-284.

S. M. Weiss and C. A. Kulikowski (1991). *Computer Systems that Learn. Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems.* Morgan Kaufmann Publishers, Inc.