Evolution of Fuzzy Rule Based Classifiers

Jonatan Gomez

Universidad Nacional de Colombia and The University of Memphis
jgomezpe@unal.edu.co, jgomez@memphis.edu

Abstract. The paper presents an evolutionary approach for generating fuzzy rule based classifier. First, a classification problem is divided into several two-class problems following a fuzzy unordered class binarization scheme; next, a fuzzy rule is evolved (not only the condition but the fuzzy sets are evolved (tuned) too) for each two-class problem using a Michigan iterative learning approach; finally, the evolved fuzzy rules are integrated using the fuzzy round robin class binarization scheme. In particular, heaps encoding scheme is used for evolving the fuzzy rules along with a set of special genetic operators (variable length crossover, gene addition and gene deletion). Experiments are conducted with different public available data sets.

Keywords: Fuzzy Rule Evolution, Fuzzy Set Tuning, Evolutionary Algorithm, Fuzzy Class Binarization

1 Introduction

Classification is a supervised learning technique that takes labeled data samples and generates a model (classifier) that classifies new data samples in different predefined groups or classes [1]. Classification has been extensively studied in machine learning and data mining [1,2,3], and has received particular attention of soft-computing techniques such as fuzzy Logic [4,5,6] and evolutionary algorithms [7,8,9]. Due to high interpretability of fuzzy rule based classifiers (**FRBC**) and the ability of evolutionary algorithms (**EA**) to find good solutions, some research work has focused on developing evolutionary techniques for generating FRBC [10,11,7,12,6,9]. These techniques receive the name of Genetic Fuzzy Rule Based Systems (**GFRBS**) [13,14]. Of several GFRBS approaches proposed, all of them differ from each other in at least one of the following aspects: number of fuzzy rules that each individual encodes, type of rule expression encoded by an individual, and scope of the evolutionary process [10,15,13].

1.1 Michigan and Pittsburgh

According to the number of crisp/fuzzy rules that each individual of the population encodes, GFRBS can be divided in two broad approaches: Pittsburgh and Michigan. Each one has its advantages and disadvantages [15]. In the Pittsburgh approach, each individual of the population encodes a set of rules that will compose the RBC [16,17,18]. It is possible to capture the rules interaction

K. Deb et al. (Eds.): GECCO 2004, LNCS 3102, pp. 1150–1161, 2004.

in the fitness function but the search space grows exponentially with respect to the number of rules encoded. In the Michigan approach, each individual of the population encodes one rule [19,20]. Although the search space is small compared with the Pittsburgh approach, only one rule is encoded, it is not possible to capture the rule interactions in the fitness function [15]. Michigan approaches are further divided in two groups: simple and iterative learning. In the simple approach, all the rules are evolved using a single EA run. It is done by introducing a niching strategy in the EA [21,22,23]. In the iterative learning approach, the set of rules is evolved in several runs of an EA - a rule is evolved in each run [12,7,6]. The number of EA runs, the type of rule evolved and the mechanism used for combining such rules depends on the particular approach [15]. Some approaches penalize rules evolved in previous iterations and stop the iterative process when the set of rules is adequate [12,6,13,24]. Other approaches run an EA as many times as the number of classes the problem has, each run with a different target class [25].

1.2 Rule Encoding

Several fuzzy rule encoding mechanisms have been proposed for GFRBS. Some of them are:

- **Conjunctions of simple terms**. The condition length is fixed. It is composed by atomic expressions connected with a fuzzy *and* logic operator. Such atomic expressions are the only elements evolved [11,7,12,6,9].
- **Fixed condition structure.** The condition is determined by a template where the logic operators and the tree structure are fixed. The atomic conditions are the only elements evolved [8].
- **Linear-tree representation with precedence of operators.** The tree structure of the condition is determined by priorities associated with each logic operator in the condition. Atomic expressions, logic operators and operator priorities are all evolved [25].
- **Heaps or complete binary tree structures.** The tree structure of the condition is always a heap (binary trees filled by levels from left to right). Atomic expressions and logic operators are evolved [10].

1.3 Evolution Scope

It is possible to use an EA for evolving (tuning) the fuzzy sets membership functions at the same time the fuzzy rule is evolved [7,26,14]. In [27], Murata proposes a binary encoding for evolving fuzzy sets in such a way that a bit on '1' indicates when a fuzzy set membership has the value 1.0. The closest bits in on, at the left and right of such bit, indicate that the fuzzy set membership has value 0.0. In this way, fuzzy sets are encoded along with the fuzzy rule being evolved. Karr proposed a mechanism for evolving triangular fuzzy sets in [28]. Karr encoded into the chromosome the two control points that define the base of the triangular fuzzy set. The highest point, the point that will take the maximum fitness value of 1.0, is defined as the middle point between the evolved control points. Each control point is encoded independently using a set of m bits.

This paper provides a fuzzy set tuning mechanism to the heap encoding scheme proposed by Gomez et al. in [10], simplifies the fitness function, and uses a fuzzy unordered class binarization scheme to divide a multi-class problem into several two class problems. This paper is divided in four sections. Section 2 describes the proposed approach: fuzzy rule encoding, fuzzy set tuning mechanism, fitness function and class binarization. Section 3 presents the experiments performed and the analysis of results. Section 4 draws some conclusions.

2 Proposed Approach

In order to evolve a FRBC, we developed a fuzzy round robin binarization technique and used an evolutionary algorithm (EA) for evolving a fuzzy rule for each two-class classification problem (Michigan approach). The EA takes as input the training data set (preprocessed to represent only two classes), applies the evolutionary strategies, and returns one fuzzy rule. Such a rule has the form : R: **IF** *condition* **THEN** data is *positive*. A data sample is classified as positive with the truth-value (**TV**) of the fuzzy rule R and classified as negative with TV equal to the fuzzy negation of the TV of R.

2.1 Fuzzy Unordered Class Binarization

An unordered class binarization transforms an m-class problem into m two-class problems, where the i-th classifier that is generated using the samples of class i as positive samples and samples of the other classes *(j=1..m, j≠i)* as negative samples [29]. Each classifier is generated using only samples of the two corresponding classes. Algorithm 1 presents a fuzzy version of unordered class binarization. This binarization scheme has been successfully applied with a maximum defuzzyfication technique in [10,30,25].

Algorithm 1 Fuzzy Unordered Classification

CLASSIFY($classifier[1..m]$, $sample$)
1. $winners = \emptyset$
2. **for** $i = 1$ **to** m **do**
3. $winners = winners \cup \{ \mu_{positive}(classifier_i, sample) \}$
4. **return** DEFUZZY($winners$)

2.2 Fuzzy Rule Encoding

Because an evolutionary algorithm is executed for each two class problem, and a single fuzzy rule is the classifier associated to it, it is not necessary to encode the class that the fuzzy rule is discriminating (it is always the positive class). Only the fuzzy expression that corresponds with the condition part of the fuzzy rule is encoded. In this paper, we extend the heap encoding scheme proposed by

Gomez et al. in [10] by allowing the EA to evolve the fuzzy sets associated with the atomic conditions. A heap tree is a binary tree that is filled completely on all the levels except possibly the last level that is filled from left to right [31], see figure 1.

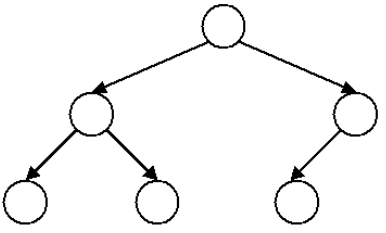


Fig. 1. Heap.

Gomez et al. [10] shown that it is possible to use a linear structure (individual chromosome) for representing such heap expression trees. This structure is defined as a list of genes, each gene encoding an atomic expression (defined as *fuzzy_variable [is/not] fuzzy_set*) and a logic operator (*and* or *or*). The logic operator encoded in the last gene is not taken into account because the number of logic operators is one less than the number of atomic expressions, see Figure 2.

$Gene_1$				...	$Gene_{n-1}$				$Gene_n$			
$Atom_1$			Op_1	...	$Atom_{n-1}$			Op_{n-1}	$Atom_n$			*
var	$\in$ / $\notin$	set	$\wedge/\vee$	...								*

Fig. 2. Linear Representation of heaps.

Given a chromosome with n genes, $A = a_1a_2..a_n$, the encoded heap expression can be obtained using 1.

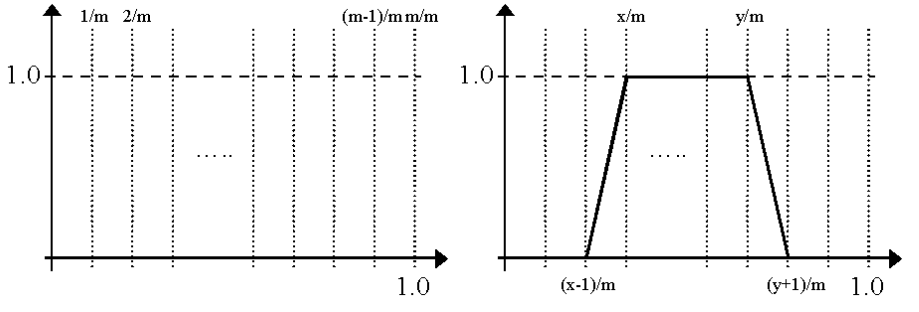
$$Tr\left(a_1a_2..a_n\right) = \begin{cases} [\lambda,\, atomic\left(a_1\right),\, \lambda] & if\, n = 1 \\ rep\left(Tr\left(a_1a_2..a_{n-1}\right), atomic\left(a_n\right), oper\left(a_n\right)\right) & other\, case \end{cases} \tag{1}$$

Here, $rep\left(T,\, A,\, O\right)$ replaces the first leaf node of T (using the level tree enumeration [31]), with the node $[first_{leaf}\left(T\right),\, O,\, A]$.

We use $\lceil \log_2\left(m\right) \rceil$ bits for encoding m possible attributes, one bit for the membership relation ($\in$ / $\notin$) and one bit for the logic operator ($\wedge/\vee$). The number of bits used for representing the fuzzy set depends on the scope of the evolutionary process; whether it is a fixed fuzzy space or fuzzy set tuning.

2.3 Fuzzy Set Tuning

Instead of encoding the index of a predefined fuzzy set into each gene as proposed by Gomez et al. in [10] where $\lceil \log_2 (m) \rceil$ bits are used for representing m predefined fuzzy sets, a set of parameters defining a fuzzy set can be encoded into each gene and allow the EA to tune it. Isosceles triangular fuzzy sets can be tuned by encoding two values, the points defining the base of the triangle [28]. Gaussian shaped fuzzy sets can be defined by encoding two values, the median and the standard deviation. In this paper, the fuzzy set tuning is restricted to trapezoidal fuzzy sets defined by two parameters. The attribute space is divided into m regions of the same length (m is a parameter given by the user). This division generates $m+1$ control points, see Figure 3.a. Given two control points, x and y ($x \leq y$), it is possible to define the trapezoidal fuzzy set: ($max\left\{0, \frac{x-1}{m}\right\}$, $\frac{x}{m}$, $\frac{y}{m}$, $min\left\{1, \frac{y+1}{m}\right\}$), see Figure 3.b[1].



(a) Division of the space in m intervals (b) Two control points trapezoid

Fig. 3. Tuning of Trapezoidal membership functions.

Therefore, $2\lceil \log_2 (m+1) \rceil$ bits are used for representing the two control points of the trapezoidal fuzzy set.

2.4 Genetic Operators

Variable Length Simple Point Crossover (VLSPX). Given two chromosomes $A = a_1 a_2..a_n$ and $B = b_1 b_2..b_m$, n and m are the size in bits of A and B respectively, the VLSPX selects a random point k in the interval $[2, min\{n, m\} - 1]$, and generates two offspring $C = a_1 a_2..a_k b_{k+1}..b_m$ and $D = b_1 b_2..b_k a_{k+1}..a_n$. When PFEs are encoded, it is possible that VLSPX does not only exchange genes but modifies one of them (if the crossover point is selected in the middle of such gene).

[1] It will define a triangular fuzzy set instead of a trapezoidal if $x = y$.

Single Bit Mutation (SBM). Given a chromosome $A = a_1 a_2..a_k..a_n$, the SBM produces an offspring by flipping one random bit in it, $C = a_1 a_2..\overline{a_k}..a_n$.

Gene Addition (ADD). Given a chromosome $A = a_1 a_2..a_n$ of n genes, the ADD operator produces an offspring by generating a random gene r and appending it to the end of the chromosome: $C = a_1 a_2..a_n r$.

Gene Deletion (DEL). Given a chromosome $A = a_1 a_2..a_n$ of $n \geq 2$ genes, the DEL operator produces an offspring by removing the last of the chromosome, $C = a_1 a_2..a_{n-1}$.

2.5 Fitness Function

The concept of fuzzy confusion matrix was introduced in [30] in order to determine the performance of an individual. This section proposes a simplification of such fitness function that has shown good performance in the classification problem.

Fuzzy Confusion Matrix. Values in a confusion matrix correspond with the cardinality of intersection sets. For example, PP is the number of positive samples that are classified (predicted) as positive, i.e., the cardinality of the set: Actual-Positive ∩ Predicted-Positive. These values can be calculated by using the membership function of the data samples to the actual and predicted data sets as:

$$PP = \sum_{i=1}^{n} \mu_A(d_i) \wedge \mu_B(d_i) \tag{2}$$

$$PN = \sum_{i=1}^{n} \mu_C(d_i) \wedge \mu_B(d_i) = \sum_{i=1}^{n} \overline{\mu_A(d_i)} \wedge \mu_B(d_i) \tag{3}$$

$$NP = \sum_{i=1}^{n} \mu_A(d_i) \wedge \mu_D(d_i) = \sum_{i=1}^{n} \mu_A(d_i) \wedge \overline{\mu_B(d_i)} \tag{4}$$

$$NN = \sum_{i=1}^{n} \mu_C(d_i) \wedge \mu_D(d_i) = \sum_{i=1}^{n} \overline{\mu_A(d_i)} \wedge \overline{\mu_B(d_i)} \tag{5}$$

Where, n is the number of samples used to test the classifier, A is the actual (real) positive set, B is the predicted positive set, C is the actual negative set, D is the predicted negative set, and d_i is the i-th data record sample in the data set.

Notice that, for a two-class classification problem, one only needs to know the membership of a data sample to the actual data set and to the predicted membership value of the positive set.

Equations 2, 3, 4 and 5 can be extended to fuzzy sets and fuzzy rules. The degree of membership of the data sample to the actual positive set is given by the data sample label and the predicted membership to the positive set is calculated as the truth-value of the condition part of the fuzzy rule. The confusion matrix generated by using these extensions is called **fuzzy confusion matrix**. Performance metrics like accuracy and true positives can be generated from the fuzzy confusion matrix. Such new performance metrics will be called **fuzzy performance metrics:** (fuzzy accuracy, fuzzy true positives**,** etc).

Definition. Since the goal of the evolutionary process is to generate a simple fuzzy rule that can discriminate the positive class from the negative, the fitness of an individual is defined by the fuzzy accuracy (**FAC**), and the fuzzy rule length (**FRL**), i.e. the number of atomic conditions defining the fuzzy rule. In this way, the optimization problem is a two-goal objective function: maximizing the FAC while minimizing the fuzzy rule length (FRL). Although there are several ways to deal with multi-goal objective functions, in this work, the weighted sum technique was used. Therefore, the fitness of an individual is calculated using equation 6.

$$\begin{aligned} fitness(R) &= w * FAC(R) + (1-w) * \left(1 - \frac{FRL(R)}{M}\right) \\ &= w * \left(\frac{PP(R)+NN(R)}{PP(R)+PN(R)+NP(R)+NN(R)}\right) + (1-w) * \left(1 - \frac{FRL(R)}{M}\right) \end{aligned} \tag{6}$$

Here, w is the weight associated with the fuzzy accuracy reached by the individual and M is the maximum number of atomic expressions defining a fuzzy rule.

2.6 Rule Extraction

The best individual of the population, according to the fitness value, will determine the fuzzy rule that will be used for discriminating between the two classes under consideration.

3 Experimentation

3.1 Experimental Settings

Five benchmark data sets (publically available), were used as a test bed. See table 1. A 10-fold cross-validation was applied to each data set 5 different times. The reported results are the average over those 50 runs.

For each data set, we used the Hybrid Adaptive Evolutionary Algorithm (**HaEa**) proposed by Gomez [32] as the two-class evolutionary algorithm (which evolves the fuzzy rule associated with each two-class problem) . HaEa adapts the genetic operator rates while evolves the solution of the problem. HaEa was executed for 100 iterations using 100 individuals as population and VLSPX,

Table 1. Test bed

DATA SET	CLASSES	DIM	SAMPLES	
			TOTAL	PER CLASS
BREAST	2	9	699	{458, 241}
PIMA	2	8	768	{500, 268}
HEART	2	13	270	{150,120}
IRIS	3	4	150	{50, 50, 50}
WINE	3	13	178	{59, 71,4 8}

SBM, ADD and DEL as genetic operators. Individuals of the initial population were randomly generated with a length varying between 1 and the number of attributes defining the data set (9 for *Breast*, 8 for *Pima* and 13 for *Heart*). A 10 fold cross-validation technique was applied to each data set 5 different times. The reported results are the average over those 50 runs. We used the *average-and* ($TV\,(p \wedge q) = \frac{2*TV(p)*TV(q)}{TV(p)+TV(q)}$) as fuzzy *and* operator[2], *max* as fuzzy *or* operator, and $1.0 - x$ as fuzzy *not* operator. We compared the performance of the fuzzy rules evolved using the fuzzy set tuning mechanism (with 6 divisions) against the fuzzy rules evolved using a fixed collection of 5 well tuned fuzzy sets, as shown in figure 4. We set the number of division to 6 in order to match the division generated using the 5 predefined fuzzy sets.

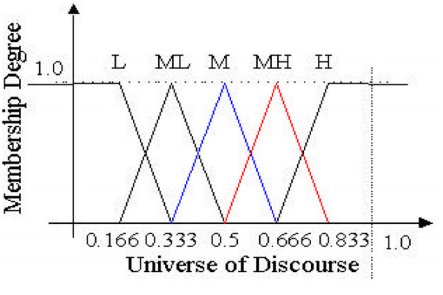


Fig. 4. Fixed Collection of Fuzzy Sets

3.2 Results and Analysis

Comparing fuzzy sets tuning against fixed fuzzy sets. Table 2 shows the performance of the fixed and tuning fuzzy sets approach.

[2] The *average-and* fuzzy logic operator produced better results than the *min-and* operator.

Table 2. Performance of fuzzy set tuning against fixed fuzzy sets.

	PRE-DEFINED	TUNING
BREAST	94.94±2.71	**94.85±2.41**
PIMA	73.79±5.75	**74.21±5.63**
HEART	78.44±7.79	**78.96±8.26**
IRIS	94.51±4.83	**95.20±5.97**
WINE	92.78±6.13	**93.18±6.72**

As shown, the performance reached by the proposed approach using tuning of fuzzy sets is better than the performance reached using a predefined collection of fuzzy sets in almost all the data sets (exception done with the *Breast* data set). These results indicate that the tuning mechanism can evolve fuzzy sets that approximate patterns hidden in the data set. Take for example the following fuzzy rule generated for the *Pima* data set in a sample run:

IF x_8 *is* $set_{2,3}$ ***AND*** x_1 *is* $set_{0,1}$ ***OR*** x_2 *is not* $set_{0,4}$ ***THEN*** *Diabetes-Disease*

Here, $set_{x,y}$ represents the trapezoidal fuzzy set:

$$(max\left\{0,\ \tfrac{x-1}{m}\right\},\ \tfrac{x}{m},\ \tfrac{y}{m},\ min\left\{1,\ \tfrac{y+1}{m}\right\})$$

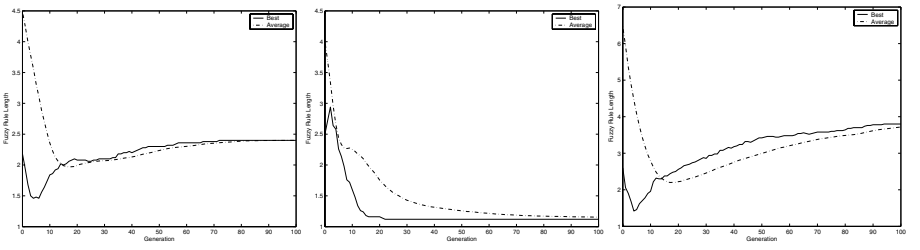
with m being the number of divisions. In order to approximate the atomic expression $x_8\, is\, set_{2,3}$ using a collection of predefined fuzzy sets, it will require other type of fuzzy or logic operator, like *Restricted-Sum Or,* and the condition $x_8\, is\, ML$ ***OR*** x_8 *is* M.

Fuzzy Rule Complexity. Figure 5 shows the evolution of the fuzzy rule length for both the best individual in the population and the average length of individuals in the population. Clearly, our approach using the tuning mechanism produces simple fuzzy rules as no more than 4 attributes are included in the condition part of the fuzzy rules.

Comparison with Results Reported in the Literature. We took the results produced by our approach, Tuning of fuzzy sets with Heaps encoding (**T-HEAP**), and compared them against results reported in the literature. See table 3[3]. As shown, our results (first row) compare well[4]

[3] Results reported for QDA, LDA, C4.5, kNN, SSV and FSM taken from [33]. Results for GAP taken from [27], where the number of fuzzy rules was close to the number of classes. Results for CTree are taken from [10].

[4] Although all of these results were obtained with different statistical validation methods (leave-one-out, or 10-cross-validation) or not statistical validation, the values reported here are an indicative of the performance of the proposed approach.



(a) (b) (c)

Fig. 5. Fuzzy Rule Length Evolution. (a) BREAST, (b) PIMA, (c) HEART.

Table 3. Comparative performance of the proposed approach

Method	BREAST	PIMA	HEART	WINE	Statistical Test
T-HEAPS	**94.85**	**74.21**	**78.96**	**93.18**	**10-cross-validation**
QDA	94.90	74.80	57.80	99.40	Leave-one-out
LDA	96.00	77.20	60.40	98.90	Leave-one-out
GAP-sel	-	-	-	97.20	None
GAP-par	-	-	-	93.30	None
C4.5	94.70	73.00	22.90	-	Leave-one-out
kNN	96.90	71.90	65.60	95.50	Leave-one-out
SSV	96.30	73.70	-	98.30	10-cross-validation
FSM	96.90	-	-	96.10	10-cross-validation
WM	87.10	71.30	-	-	-
GIL	90.10	73.10	-	-	-
ABD	96.00	75.90	-	-	-
ABA	95.10	74.80	-	-	random (50-50)%

4 Conclusions

We proposed a technique for evolving fuzzy rules that follows an iterative Michigan approach. The iterative process is performed using a fuzzy unordered class binarization scheme. A fuzzy set tuning mechanism was developed for the Heaps encoding strategy proposed by Gomez et al. in [10]. Also, a simplified fitness function was introduced. The results obtained indicate that the proposed approach is able to evolve good fuzzy rule based classifiers. In general, the quality of such evolved classifiers is higher when the fuzzy set tuning scheme was included in the evolutionary process.

References

1. J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.
2. R. S. Michalski, I. Bratko, and M. Kubat, *Machine Learning and Data Mining: Methods and Applications*. J. Wiley & Sons, 1998.
3. R. Holte, "Very simple classification rules perform well in most common used datasets," *Machine Learning*, no. 11, pp. 63–91, 1993.
4. Y.-C. Hu, R.-S. Chen, and T. G-H., "Finding fuzzy classification rules using data mining techniques," *Pattern Recognition Letters*, no. 24, pp. 509–519, 2003.
5. Q. Shen and A. Chouchoulas, "A rough-fuzzy approach for generating classification rules," *Pattern Recognition*, no. 35, pp. 2425–2438, 2002.
6. A. Gonzalez and R. Prez, "Completeness and consistency conditions for learning fuzzy rules," *Fuzzy Sets and Systems*, no. 96, pp. 37–51, 1998.
7. H. Ishibushi and T. Nakashima, "Liguistic rule extraction by genetics-based machine learning," in *Proceedings of the Genetic and Evolutionary Computation Conference GECCO'00*, pp. 195–202, 2000.
8. A. Giordana and L. Saitta, "Regal: An integrated system for learning relations using genetic algorithms," in *Proceedings of the Second International Workshop on Multi-strategy Learning*, pp. 234–249, 1993.
9. K. De Jong and W. Spears, "Learning concept classification rules using genetic algorithms," in *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pp. 651–656, 1991.
10. J. Gomez, D. Dasgupta, O. Nasraoui, and F. Gonzalez, "Complete expression trees for evolving fuzzy classifier systems with genetic algorithms," in *Proceedings of the North American Fuzzy Information Processing Society Conference NAFIPS-FLINTS 2002*, pp. 469–474, 2002.
11. M. V. Fidelis, H. S. Lopes, and A. A. Freitas, "Discovering comprehensible classification rules with a genetic algorithm," in *Proceedings of Congress on Evolutionary Computation (CEC)*, pp. 805–810, 2000.
12. J. Liu and J. Kwok, "An extended genetic rule genetic algorithm," in *Proceedings of Congress on Evolutionary Computation (CEC)*, pp. 458–263, 2000.
13. O. Cordon, A. Gonzalez, F. Herrera, and R. Perez, "Encouraging cooperation in the genetic iterative rule learning approach for quality modeling," in *Computing with Words in Intelligent/Information Systems 2. Applications, J. Kacprzyk, L. Zadeh (Eds.)*, Physica-Verlag, 1998.
14. O. Cordon and F. Herrera, "A general study on genetic fuzzy system," in *Genetic Algorithms in Engineering and Computer Sciences*, pp. 33–57, Jonh Wiley and Sons, 1995.
15. A. A. Freitas, "A survey of evolutionary algorithms for data mining and knowledge discovering," in *Advances in Evolutionary Computation. A. Ghosh and S. Tsutsui. (Eds.)*, Springer-Verlag, 2001.
16. K. De Jong, W. Spears, and D. F. Gordon, "Using genetic algorithms for concept learning," *Machine Learning Research*, no. 13, pp. 161–188, 1993.
17. C. Z. Janikow, "A knowledge-intensive genetic algorithm for supervised learning," *Machine Learning Research*, no. 13, pp. 189–228, 1993.
18. S. F. Smith, *A Learning System based on Genetic Adaptive Algorithms*. Ph. D. Thesis, University of Pittsburgh, 1980.
19. G. Giordana and F. Neri, "Search-intensive concept induction," *Evolutionary Computation*, no. 3(4), pp. 375–416, 1995.

20. L. Booker, *Intelligent Behaviour as an Adaption to the Task Environment*. Ph. D. Thesis, University of Michigan, 1982.
21. S. W. Mahfoud, "Crowding and preselection revisited," in *Proceedings Second Conference Parallel Problem Solving from Nature*, 1992.
22. D. Goldberg and J. J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," in *Proceedings Second International Conference on Genetic Algorithm*, pp. 41–49, 1987.
23. J. H. Holland, *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1975.
24. O. Cordon and F. Herrera, "A three-stage evolutionary process for learning descriptive and approximate fuzzy logic controller knowledge bases," *International Journal of Approximate Reasoning*, no. 17(4), pp. 369–407, 1997.
25. D. Dasgupta and F. Gonzalez, "Evolving complex fuzzy classifier rules using a linear tree genetic algorithm," in *Proceedings of the Genetic and Evolutionary Computation Conference GECCO'01*, pp. 299–305, 2001.
26. B. Carse, T. Fogarty, and A. Munro, "Evolving fuzzy rule based controllers using genetic algorithms," *Fuzzy Sets and Systems*, no. 80, pp. 273–294, 1996.
27. T. Murata, S. Kawakami, H. Nozawa, M. Gen, and H. Ishibushi, "Three-objective genetic algorithms for designing compact fuzzy rule-based systems for pattern classification problems," in *Proceedings of the Genetic and Evolutionary Computation Conference GECCO'01*, pp. 485–492, 2001.
28. C. Karr, "Genetic algorithms for fuzzy controllers," *AI Experts*, pp. 26–33, 1991.
29. J. Fürnkranz, "Round robin classification," *Machine Learning Research*, no. 2, pp. 721–747, 2002.
30. J. Gomez and D. Dasgupta, "Evolving fuzzy rules for intrusion detection," in *Proceedings of the Third Annual IEEE Information Assurance Workshop 2002 Conference*, pp. 68–75, 2002.
31. T. Cormer, C. Leiserson, and R. Rivest, *Introduction to Algorithms*. McGraw Hill, 1990.
32. J. Gomez, "Self adaptation of operator rates in evolutionary algorithms," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2004)*, June 2004.
33. D. Wlodzislaw, "Data sets used for classification: Comparison of results," in *http://www.phys.uni.torun.pl/kmk/projects/datasets.html*.