18th International Conference of the North American Fuzzy Information Processing Society – NAFIPS

June 10-12, 1999
New York, New York, U.S.A.

Edited by

Rajesh N. Davé
Thomas Sudkamp

Sponsored by

NAFIPS - North American Fuzzy Information Processing Society

In Cooperation with

IEEE Neural Networks Council
IEEE Systems, Man and Cybernetics Society

Organizing Institution

New Jersey Institute of Technology, USA

99TH8397

A Fuzzy Classifier using Genetic Algorithms for Biological Data

Jim Diederich
University of California
Department of Mathematics
Davis, CA 95616
dieder@math.ucdavis.edu

Renaud Fortuner
Scientific consultant
Correspondant du Muséum National d'Histoire
Naturelle, Paris
86420 Monts sur Guesnes, France
fortuner@wanadoo.fr

Abstract

This paper presents a fuzzy classifier that generates fuzzy if-then rules from numerical data for multi-dimensional classification problems using genetic algorithms with rejection of low confidence patterns.

1. Introduction

Ishibuchi and his colleagues have described various fuzzy classifiers for crisp numerical training data for pattern classification problems; see for example [1,2,3]. When the problem exceeds more than a handful of attributes, they use genetic based algorithms [3] to search the large space of fuzzy rules to produce small rule sets with very high classification rates. Rejection options are not considered. However, if training data with low confidence levels are rejected, then their genetic based method may result in a significant percentage of rejects. In this paper we modify the methods in [3] to reduce the number of rejects without artificially inflating the confidence levels of boundary data.

For two closely related, well described species of nematodes, i.e., microscopic round worms, distinguishing between unknowns can be very difficult and may ultimately rely on quantitative data, typically measurements. In [4], Fortuner studied two such species, Helicotylenchus dihystera, H.d., and Helicotylenchus pseudorobustus, H.p. We will use the data from [4] for 23 populations of H.d. and 15 populations of H.p. for our training data. Measurements for eight characters (attributes) are considered for each population: LON, the length of the body; OES, the length of the oesophagus; STY, the length of the stylet; PHAS, the position of the phasmides; PRO, the length of the process terminal; QUE, the length of the tail; DVU, the diameter of the body; TAN, the number of tail annuli. The data for the first seven of these can be found in [4], pp. 194-222; that for PRO is unpublished. For each of these nematode populations and each attribute, the mean and the range values are available. When we train with crisp data, the mean alone is used and when we train with fuzzy data, the value is represented by a triangular membership function, with the vertex at the mean, with membership value 1.0, and the range for its support. As in [5], we will also compare using crisp and fuzzy data for training.

2. Background

This paper extends the methods of [3] by incorporating concepts from [5]. We simplify the exposition by assuming a 2-class problem, with classes Cl-1 and Cl-2, and n attributes for each training point (pattern). In this section we present the methods of [3] simplified to the 2-class problem. The end of each procedure will be marked by a ‡. TP will denote a collection of m training patterns $x_p = (x_{p1}, ..., x_{pn})$, where, for this section, x_{pi} is the (crisp) value of its i^{th} attribute. Each domain is normalized and uniformly partitioned to create K membership functions representing a fuzzy grid for the domain. Using K = 1, ... , 6 gives a total of 21 grid functions as seen in Fig. 1; let G denote this collection.

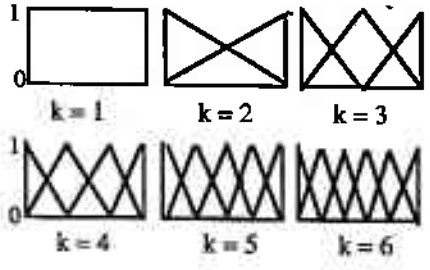

Figure 1. The collection G.

In our tests we excluded the k = 1 grid function from G. A member of the grid is denoted $A_{j\,i}$, where i is the index of the attribute, and μ_{ji} is its membership function. We assume rules of the form:

0-7803-5211-4/99/$10.00 © 1999 IEEE

Rule R_j: If x_{p1} is A_{j1} and ... and x_{pn} is A_{jn}

then classify x_p as Class C_j with $CF = CF_j$

where $A_{ji} \varepsilon G$, C_j is the consequent class, and CF_j is the grade of certainty.

Initial Rule Generation. With large n, even n = 8, it is impractical to generate all possible rules. Therefore, some small subset must be selected, typically 100 or fewer [3].

PROCEDURE IRG:

Step 1. Specify N_{pop}, the rulebase size.

Step 2. Create the antecedents for rule R_j, $j = 1, \ldots, N_{pop}$, by randomly selecting A_{ji}'s from G.

Step 3. Complete the generation of each rule R_j, $j = 1, \ldots, N_{pop}$, using Proc. RG (presented next.) ‡

Rule Generation. A fuzzy rule R_j with antecedents A_{ji} can be generated by the next method that computes the consequent, Cj, of the rule and its certainty factor, CFj.

PROCEDURE RG:

Step 1. Calculate for each class Cl-i, i = 1, 2

$$\beta_i = \sum_{x_p \varepsilon \text{ Cl-i}} U_j(x_p)$$

where the compatibility grade is

$$U_j(x_p) = \mu_{j1}(x_{p1}) \cdot \ \ldots \ \cdot \mu_{jn}(x_{pn}).$$

Step 2. The max $\{\beta_1, \beta_2\}$ determines the class C_j of rule R_j according to the largest β if the β_i's are unequal, otherwise the class is undetermined.

Step 3. If a class is determined in Step 2, then set

$$CF_j = |\beta 1 - \beta 2| / |\beta 1 + \beta 2|. \ ‡$$

Fuzzy Reasoning. This describes how the class of a pattern x_p can be determined if its class is unknown, or for classifying a training pattern by assuming its class is unknown. Assume that S is a set of fuzzy if-then rules.

PROCEDURE FR:

Step 1. For an x_p, calculate for each i = 1, 2

$$\alpha_i = \max_j \{ U_j(x_p) \cdot CF_j \mid C_j = \text{Cl-i and } R_j \varepsilon S \}$$

Step 2. $\alpha_p = \max\{\alpha_1, \alpha_2\}$ determines of class of x_p to be Cl-i if $\alpha_p = \alpha_i$ and $\alpha_1 \neq \alpha_2$. Otherwise, the class is not determined. ‡

In [3], where genetic methods are used, a confidence level is not computed as it is in [1,2] or in [6] where it is used as a reject option. The next method computes confidence levels after Steps 1 & 2 in Proc. FR.

PROCEDURE FRC:

Steps 1 & 2 as in Proc. FR.

Step 3. If a class is determined in Step 2, compute the confidence level $\sigma_p = \alpha_p - \min\{\alpha_1, \alpha_2\}$. ‡

Learning Algorithm. For the current population of rules, learning can be done by rewarding rules when they correctly identify a pattern and punishing them when they don't.

PROCEDURE LA:

For a current population of rules S repeat N_{learn} times:

For p = 1 to m

Step 1. Take x_p in TP and determine the rule R_j in S that yields the value for α_p in Proc. FR (or Proc. FRC).

Step 2. If x_p is correctly identified, increase CFj

$$CF_j := CF_j + \eta_1 (1 - CF_j),$$

otherwise decrease it

$$CF_j := CF_j - \eta_2 \, CF_j,$$

where η_1 and η_2 are positive-valued learning rates. ‡

Genetic Changes. In order to obtain the next generation of rules from the current population S using genetic methods, a fitness measure is needed. For each rule R_j in S, its fitness is defined by [3]

$$\text{fitness}(R_j) = w_{NCP} \cdot NCP(R_j) - w_{NMP} \cdot NPM(R_j),$$

where $NCP(R_j)$ is the number of points correctly classified by R_j. $NPM(R_j)$ the number incorrectly classified, and w_{NCP} and w_{NMP} are positive weights. Let $\text{fitness}_{min}(S)$ be the smallest fitness value for all rules.

PROCEDURE GC:

Step 1. Compute the fitness of each rule in S.

Step 2. Compute selection probabilities for each rule R_j

$$P(R_j) = \frac{\text{fitness}(R_j) - \text{fitness}_{min}(S)}{\sum \{\text{fitness}(R_k) - \text{fitness}_{min}(S)\}}$$

where the sum is over all rules in S.

Step 3. Select $S \cdot P_{rep}$ rules, collected in pairs, according to the probabilities $P(R_j)$, where P_{rep} is the proportion of rules to be replaced.

Step 4. Replace the $S \cdot P_{rep}$ rules with the lowest fitness values by the pairs from Step 3 that are first modified by crossover and mutation. In crossover, antecedents of the rules in each pair are swapped using a uniform probability. In mutation, according to a mutation rate, antecedents are randomly replaced by grid functions randomly selected from G. The consequent class C_j and the certainty factor CF_j of the replacement rules are computed as in Proc. RG. ‡

Genetic Training. The methods presented above are used to construct the main training method of [3], which is:

PROCEDURE Gen-1:

Step 1. Proc. IRG "Initialize the rulebase S"
Step 2. Proc. LA "Perform learning"
Step 3. Proc. FR on all patterns in TP to obtain
Tot = number correctly classified.
Step 4. Proc. GC "Modify S for next generation"
Step 5. Until N_{gen} generations are done, go to Step 2.
Step 6. Output S with highest Tot. ‡

Instead of using Proc. Gen-1 to compare with our methods, we will use a slight modification. We wish not only to maximize the total correctly identified but also the total confidence level. The confidence levels σ_p convey for each pattern the degree to which the system can confirm a classification. They can also be used for rejecting patterns with low confidence values [6]. Consequently, we replace Tot in Proc. Gen-1 by

$$Tot_c = Tot + \sum_p \underline{\sigma}_p,$$

where $\underline{\sigma}_p = \sigma_p$ if x_p is correctly identified, and $\underline{\sigma}_p = -\sigma_p$ if it is not. We could compute Tot_c by omitting rather than subtracting confidence levels of incorrectly identified patterns. Our test results would not be significantly different.

PROCEDURE Gen-2:

In Proc. Gen-1 use Proc. FRC instead of Proc. FR in Step 3. Use Tot_c in place of Tot. ‡

3. Main Methods

In this section we propose modifications to the methods in Section 2. This will be based on the 3-way ordering method in [5]. We propose two essential changes to Proc. Gen-2: first, Proc. IRG is modified to produce an initial set of rules up to half of which are generated using 3-way ordering, with the remainder randomly generated; second, the compatibility grade $U_j(x_p)$ used in Proc. RG and Proc. FRC is modified.

To use 3-way ordering, the attributes must be ordered and grouped three at a time, except for possibly the last group or two, which occurs when n is not a multiple of 3. Various heuristics can be employed for this, but the basic idea is to find the "best" three attributes using the training set TP, then the next best group of three and so on. For example, for the nematode data the best group of three attributes is: (PRO, STY, QUE); the next best group is (LON, DVU, PHAS); the last group is (TAN, OES). In our example, three rulebases RB1, RB2, and RB3 are then constructed using just the attributes in each group. Using only the grid for K = 6, each rulebase can contain at most $6^3 = 216$ rules, so all rules can be generated. Rules with very low CF's can be deleted. With this in mind we present our modifications to Proc. Gen-2. The first method will use 3-way ordering to generate the initial S.

PROCEDURE IRG-3-way:

Step 1. Determine a descending order for the attributes grouped three at a time as in [5] and form corresponding rulebases $RB_1, ..., RB_r$ using the largest grid size, K = 6.

Step 2. For each rulebase RB_i compute the total confidence levels $Cf_i = \sum_p \underline{\sigma}_p$ over TP and set $w_i = Cf_i / \sum_j Cf_j$.
(Note that only the attributes in the group are used to compute the compatibility grade used for σ_p.)

Step 3. Test the training patterns TP by computing for each x_p its confidence σ_{pi} for each rulebase RB_i. Store the rule R_{pi} from RB_i that produces σ_{pi}. Compute the weighted confidence $wc_p = \sum_i \underline{\sigma}_{pi} \cdot w_i$ for x_p.

Step 4. For each x_p, form a rule R_p consisting of the antecedents of the R_{pi}'s obtained in Step 3. Rank the R_p based on the value of wc_p in Step 3.

Step 5. Take the min { | TP |, $N_{pop}/2$ } best rules (ranked in Step 4) to produce up to half of the rules for the initial S. | TP | is the size of TP. The remaining rules are obtained with antecedents selected randomly, as in Proc. IRG, to complete S.

Step 6. Complete the generation of each rule Rj, j =1, ..., N_{pop}, using Proc. RG-3-way (presented next.) ‡

PROCEDURE RG-3-way:

The same as Proc. RG except for the compatibility grade $U_j(x_p)$ we use

$$U_j(x_p) = w_1 \cdot \mu_{j1}(x_{p1}) \cdot \mu_{j2}(x_{p2}) \cdot \mu_{j3}(x_{p3}) + w_2 \cdot \mu_{j4}(x_{p4}) \cdot \mu_{j5}(x_{p5}) \cdot \mu_{j6}(x_{p6}) + \ldots$$

where w_i are the weights computed in Step 2 of Proc. IRG-3-way. We assume that the attributes have been ordered and grouped. ‡

PROCEDURE FRC-3-way:

The steps are the same as Proc. FRC except for using the compatibility grade $U_j(x_p)$ from Proc. RG-3-way. ‡

With these modifications, we can state our main training method. Except for the initial rule generation and the computation of the compatibility grades, Proc. Gen-2 and Proc. Gen-3-way are the same.

PROCEDURE Gen-3-way:

Step 1. Proc. IRG-3-way "Initialize the rulebase S"
Step 2. Proc. LA "Perform learning"
Step 3. Proc. FRC-3-way on all points in TP to obtain Tot_c
Step 4. Proc. GC "Modify S for next generation"
Step 5. Until N_{gen} generations are done, go to Step 2.
Step 6. Output S with highest Tot_c ‡

4. Some variations

In this section we offer some variations on the methods described in Sections 2 and 3.

Compatibility grades. The first variation modifies the compatibility grade $U_j(x_p)$ used in Gen-2 to reduce the effect of the number of factors when the number of attributes is not small. The first variation is

PROCEDURE Gen-h/n:

Use $(U_j(x_p))^{h/n}$, where n is the number of attributes, $1 \le h \le n$, instead of $U_j(x_p)$ in Proc. Gen-2. ‡

If h = n, there is no change; if h = 1, then it is the geometric mean. In our tests we will use h = 3 and n = 8. The reason is that the 3-way calculation of the compatibility grade has three membership factors for each group of attributes. Using Gen-3/8 will give some indication of the effects of changing the compatibility grades on the number of rejected patterns.

Rule reduction. In [3] it was shown that when N_{pop} was between 70 and 100 almost all of the patterns in TP were correctly classified by approximately 20 if-then rules, where $|TP| = 150$ in three classes. We tested rule reduction in the methods presented by eliminating 2 rules from the current S for the next generation each time Tot_c exceeded the previous high. Rule reduction ceased when 50 rules was reached. The remaining generations were then computed. In our tests, 50 rules was reached when approximately half of the generations were computed. Use of rule reduction is indicated by 100↓50 in the results.

Fuzzy training patterns. All of the methods presented use crisp data. However, it is easy to accommodate x_{pi} when it is a fuzzy number, i.e., $x_{pi}(y)$ is a membership function. Instead of $\mu_{ji}(x_{pi})$ in the compatibility grade we use max $(\min(\mu_{ji}(y), x_{pi}(y)))$. Our results will indicate when crisp data is used and when fuzzy data is used.

5. Results

The results for the methods presented are given in Table 1 for the 38 patterns in the two classes, H.d. and H.p. N_{pop} was always 100 rules, except when rule reduction was performed. The other parameters used are given at the end of this section. Each method, Column 1, was run 20 times with the averages given in the table. Both crisp and fuzzy data were used as indicated in Column 2. Column 3 gives the average Tot_c. Since all 38 patterns were correctly identified in each run, the average confidence is simply $(Tot_c - 38)/38$ as shown in parentheses in Column 3. Clearly the Proc. Gen-3-way method gave much higher average confidence levels than Proc. Gen-2, and somewhat higher than Proc. Gen-3/8.

We set the rejection level at 0.1. Any pattern whose confidence level fell below this was rejected. Column 4 shows the average number of rejected points for the 20 runs and the percentage reduction for the listed method over Proc. Gen-2 for crisp and fuzzy data. It shows significant improvements using Proc. Gen-3-way.

Using reject options in genetic methods raises an interesting question. In [3], where reject options were not considered, the average percentage correctly identified was very high, essentially 100% when 70 or more rules were used. The same occurs with our data if rejection is not considered. When reject options are included, the number of rejects can vary widely from one run to the next; see Column 5. In this case one measure of the quality of a method is the number of times there are no rejects out of the 20 runs as seen in Column 6. This gives an expectation of the number of runs needed to obtain no rejects. The 3-way methods had far more instances of no rejects than the other methods.

The last column, Column 7, gives a measure of the average consistency of each method. The confidence level of a point over the 20 runs can change significantly because of the randomness that is inherent in the genetic methods. As Column 7 shows, the 3-way methods are more consistent (lower values) than the other methods, while at the same time having low numbers of rejects. The consistency for a pattern over the 20 runs is computed as the standard deviation divided by the average confidence. The values shown in Column 7 are the averages over all points.

There are two potential problems with Proc. Gen-3-way. First, the 3-way initialization of the rules could predispose

the method into finding the same local maximum. We do not believe this occurred in our tests as seen in the range of the number rejected; see Column 5. Also, the random generation of half or more of the rules in Gen-3-way helps guard against this. Second, there is a possibility that the Gen-3-way compatibility grade computation artificially inflates the confidence level of the patterns. This criticism can be leveled at the Gen-3/8 method as well. However, we observed in many runs, that points that are difficult to train in the Gen-2 method, often had the same very low confidence levels for Gen-3-way and Gen-3/8, excluding of course the runs where there were no rejects. Note that even Gen-2 produced a run with no rejects, so that the low compatibility grades caused by a large number of factors does not preclude training without rejects.

The improvement of Gen-3-way over Gen-3/8 gives some indication of how much this is due to the initial generation of the rules in Gen-3-way. The improvement of Gen-3/8 over Gen-2 gives an indication of how much a change in the compatibility grade formula contributes.

It is not clear that using fuzzy data for training in the genetic methods is an improvement over using crisp data. With the non-genetic 3-way method in [5], the rulebases constructed from fuzzy data provided more coverage of the space than the ones constructed using crisp training data and the average confidence levels were higher. In Gen-3-way the number of rules is constant for fuzzy and crisp, and the average confidence levels are not higher for fuzzy.

Finally, we note that the extra time needed to initialize S in Gen-3-way over Gen-2 is insignificant compared to the cost of producing 100 generations in each run. An improvement in time, approximately a 40% reduction, was observed in reducing incrementally the number of rules to 50 as described in Section 4. Somewhat unexpectedly, this led to even better results as seen in the last row of Table 1, which may have been due to the randomness of the rule initializations.

As in [3], the parameters used in these tests are:

$N_{pop} = 100$, rulebase size

$P_{rep} = 0.20$, rule replacement rate

$\eta_1 = 0.001$, $\eta_2 = 0.1$, learning rates)

$N_{learn} = 20$, number of learning iterations/generation

rejection level: 0.1 mutation probability: 0.1

$w_{NCP} = 1$ $w_{NMP} = 5$, rule fitness weights

$N_{gen} = 100$, number of rulebase generations/run

References

[1] Nozaki, K., Ishibuchi, H., & Tanaka, H. 1996. *Adaptive Fuzzy Rule-Based Classification Systems.* IEEE Trans. Fuzzy Systems, 4 (3): 238-249.

[2] Ishibuchi, H., Nozaki, K., & Tanaka, H. 1992. *Distributed Representation of Fuzzy Rules and Its Application to Pattern Classification.* Fuzzy Sets Sys., 52:21-32.

[3] Ishibuchi, H., Nakashima, T., & Murata, T. 1995. *A Fuzzy Classifier System That Generates Fuzzy If-Then Rules for Pattern Classification Problems.* Proc. Int. Conf. Evolutionary Comp., Perth, Australia, 2:759-764.

[4] Fortuner, R., 1987. *Variabilité et Identification des Espèces chez les Nématodes du Genre Helicotylenchus.* Éditions de l'ORSTOM, Paris.

[5] Diederich, J., & Fortuner, R. 1998. *Classification Using Small Fuzzy Biological Data Sets.* Proc. FUZZ-IEEE, Anchorage, Alaska, 1429-1434.

[6] Ishibuchi, H., & Nakashima, T. *Fuzzy Classification with Reject Options by Fuzzy If-Then Rules.* Proc. FUZZ-IEEE, Anchorage, Alaska, 1452-1457.

PROCEDURE	Data type	Average Tot_c		Average # rejected		Range # rejected	No rejects # of times	Average consistency
Gen-2	Crisp	47.3	(0.24)†	2.90		0 - 7	1	0.388
Gen-2	Fuzzy	48.0	(0.26)	6.10		2 - 10	0	0.379
Gen-3/8	Crisp	54.9	(0.44)	1.35	(53%)*	0 - 3	1	0.264
Gen-3/8	Fuzzy	52.6	(0.38)	3.55	(42%)	2 - 7	0	0.315
Gen-3/8 100↓50	Crisp	53.6	(0.41)	2.50	(14%)	1 - 6	0	0.296
Gen-3-way	Crisp	55.8	(0.47)	1.05	(64%)	0 - 3	4	0.218
Gen-3-way	Fuzzy	54.3	(0.43)	1.75	(71%)	1 - 3	0	0.215
Gen-3-way 100↓50	Crisp	55.4	(0.46)	0.80	(72%)	0 - 2	6	0.218

† average confidence level in parentheses.

* percentage reduction in *Average # rejected* for this method over Gen-2 for the same type of data.

Table 1. Results for 20 runs per procedure.