Consistent, Complete and Compact Generation of DNF-type Fuzzy Rules by a Pittsburgh-style Genetic Algorithm

Jorge Casillas and Pedro Martínez

***Abstract*— When a flexible fuzzy rule structure such as those with antecedent in conjunctive normal form is used, the interpretability of the obtained fuzzy model is significantly improved. However, some important problems appear related to the interaction among this set of rules. Indeed, it is relatively easy to get inconsistencies, lack of completeness, redundancies, etc. Mostly these properties are ignored or mildly faced. This paper, however, focuses on the design of a multiobjective genetic algorithm that properly considers all these properties thus ensuring an effective search space exploration and generation of highly legible and accurate fuzzy models.**

I. Introduction

In knowledge discovery in databases we can distinguish between two different approaches [1]: *predictive* induction and *descriptive* induction. The difference lies in the main objective pursued and, therefore, the learning method used to attain that. On the one hand, predictive induction looks for generating legible models that describe with the highest reliability the data set that represent the analyzed system. In that case, the goal is to use the obtained model to simulate the system, thus getting an explanation of its complex behavior. On the other hand, descriptive induction looks for particular (interesting) patterns of the data set. In that case, we do not get a global view of the relationships among variables but we discover a set of rules (different enough among them) statistically significant.

This paper focus on the former approach, the predictive induction, to deal with problems where both input and output are real-valued and where the knowledge obtained is important to understand better the analyzed system. To represent the knowledge, and with the aim of generating legible enough models (which, no doubt, is one of the fundamental premises in any knowledge extraction process), we propose to use fuzzy rule-based systems. These systems use IF-THEN fuzzy rules and linguistic variables to express the knowledge about the problem.

The automatic extraction of fuzzy rule-based systems can be done with different learning methods, either simple greedy algorithms [2], [3] or optimization methods such as neural networks [4], [5] and genetic algorithms (GAs) [6]. Due to the aim of this paper on generating knowledge with good interpretability, we propose to use GAs because it holds a sort of useful features for our purpose. On the one hand, they have powerful search capacity that allows us to work with multiobjective optimization. On the other, they can manage flexible representation structures mixing codification schemes or including restrictions. From the beginning of 90s many researchers have draw their attention on the use of GAs to automatically design different components of a fuzzy rule-based system [7], [8], [9]. These learning systems are usually known as genetic fuzzy systems.

Regardless the learning tool used, a crucial problem emerges: to obtain both an accurate and an understandable model. Indeed, fuzzy modeling (i.e., the process of deriving fuzzy systems) usually comes with two contradictory requirements to the obtained model: the *interpretability*, capability to express the behavior of the real system in a comprehensible way, and the *accuracy*, capability to faithfully represent the real system. Of course, the ideal thing would be to satisfy both criteria to a high degree but, since they are contradictory issues, it is generally not possible. The quest of a good trade-off between interpretability and accuracy is target of numerous research works nowadays [10], [11].

To reach this trade-off, we propose in this paper the use of fuzzy rules with antecedent in conjunctive normal form, usually know as DNF-type fuzzy rules. This representation provides a high degree of compactness and knowledge synthesis. Since we are interested on predictive induction, the Pittsburgh-style GA (where each individual encodes a complete set of rules) seems to be the best approach to properly assess the interaction among the different fuzzy rules to perform interpolative reasoning.

However, the combination of DNF-type fuzzy rules and Pittsburgh-style GA are far from be easy since several difficulties arise:

- *Consistency*: each combination of antecedent (one label per input variable) should have only one possible consequent.
- *Completeness*: every training data example should fires at least one fuzzy rule.
- *Compactness*: the lowest number of rules to accurately represent input-output relationships should be obtained. Among other issues, it involves to avoid redundant rules.
- *Non-overgeneral rules*: a DNF-type fuzzy rule should be general enough as to represent in a compact way the input-output relationship but specific enough as to avoid covering input areas without data.

Indeed, although it is relatively easy to comply with these conditions when using simple (Mamdani-style) fuzzy rules, and some authors have already faced that [12], in the case of DNF-type fuzzy rules, it becomes more difficult. Some authors have worked with DNF-type fuzzy rules [13], [14], [15], [16], [17] but they have not properly dealt with that

Jorge Casillas is with the Dept. Computer Science and Artificial Intelligence, University of Granada, E-18071 Granada, Spain (URL: http://decsai.ugr.es/~casillas).

1-4244-1210-2/07/$25.00 ©2007 IEEE.

desirable properties. Some of them use penalty fitness to correct these deficiencies, others infer a default output when no rules are fired, other tends to generate a high number of rules, some others do not take care to generate inconsistencies or redundancies... Unlike that, we propose in this paper an algorithm that *always* explores feasible solutions (according to the mentioned consistency, completeness and non-redundancy restrictions) and that guarantees a proper search orientation toward compactness and non overgeneral fuzzy rules. It considers a multiobjetive optimization process which generates a large range of solutions with different interpretability-accuracy balances under the mentioned restrictions.

The paper is organized as follows. Section II briefly presents the difficulties that appear when using DNF-type fuzzy rules. Section III describes the proposed algorithm. Section IV shows the obtained results and a comparison with other learning methods. Finally, Section V concludes and suggests some further works.

II. DNF-type Fuzzy Rule

To obtain a high degree of interpretability, we opted by a compact description based on DNF-type fuzzy rules. This kind of fuzzy rule structure has the following form:

$$\textbf{IF } X_1 \text{ is } \widetilde{A_1} \text{ and} \ldots \text{ and } X_n \text{ is } \widetilde{A_n} \textbf{ THEN } Y \text{ is } B$$

where each input variable X_i takes as a value a set of linguistic terms $\widetilde{A_i} = \{A_{i1} \vee \cdots \vee A_{il_i}\}$, whose members are joined by a disjunctive (T-conorm) operator, whilst the output variable remains a usual linguistic variable with a single label associated. The structure is a natural support to allow the absence of some input variables in each rule (simply making $\widetilde{A_i}$ to be the whole set of linguistic terms available).

However, when a whole set of such a kind of rules is learnt, collisions appear easily. Basically, these collisions are of two types.

The first kind of collision is the inconsistency. Two rules are inconsistent between them when their antecedents overlap themselves, i.e., their antecedents are the same, they coincide in some labels for each input variable, or one is subsummed by the other (i.e., an antecedent is completely contained in a larger and more comprehensive antecedent) but the consequent is different. For instance, the two following rules are inconsistent:

```
IF X1 is A1 and X2 is A2 THEN Y is C
IF X1 is A1 and X2 is A2 THEN Y is D
```

and the same in this second case where the antecedents are partially overlapped:

```
IF X1 is {A1 or B1} and X2 is A2 THEN Y is C
IF X1 is A1 and X2 is {A2 or B2} THEN Y is D
```

or in this third case where the former rule subsumes the latter:

```
IF X1 is A1 and X2 is {A2 or B2} THEN Y is C
IF X1 is A1 and X2 is A2 THEN Y is D
```

All these cases of inconsistency causes a linguistic contradiction that should be avoided.

A second, less serious problem is when the antecedent is overlapped as in any of the above examples but the consequent is the same. In that case, we have a redundancy. For example:

```
IF X1 is A1 and X2 is {A2 or B2} THEN Y is C
IF X1 is A1 and X2 is A2 THEN Y is C
```

Redundancy increases the fuzzy rule set size unnecessarily and can even provokes some undesirable interaction effects with some inference engines. When both rules have the same antecedent or one subsumes the other, the fuzzy rule set can be easily fixed by removing the repeated or the most specific rules.

The learning algorithm we propose in this paper has been designed to avoid generating solutions with inconsistencies or redundancies.

III. Pitts-DNF Algorithm

The proposed algorithm, called Pitts-DNF, has the following scheme:

```
Initialization(P)
CH = Covering_Hypermatrix(dataset)
Evaluation(P,dataset)
While (not stop condition)
   P1 = Multiobjective_Selection(P)
   P2 = Crossover(P1)
   P3 = Antecedent_Mutation(P2,CH)
   P4 = Consequent_Mutation(P3)
   P5 = Completeness_Operator(P4,dataset)
   Evaluation(P5,dataset)
   P = Multiobjective_Replacement(P5,P)
End-while
```

A. Coding scheme

Each individual of the population represents a set of fuzzy rules (i.e., Pittsburgh style). Each chromosome consists of the concatenation of a number of rules. The number of rules is not fixed *a priori* so, the chromosome size is variable-length. Each rule (part of the chromosome) is encoded by a binary string for the antecedent part and an integer coding scheme for the consequent part. The antecedent part has a size equal to the sum of the number of linguistic terms used in each input variable. The allele '1' means that the corresponding linguistic term is used in the corresponding variable. The consequent part has a size equal to the number of output variables. In that part, each gene contains the index of the linguistic term used for the corresponding output variable.

For example, assuming we have three linguistic terms (S [small], M [medium], and L [large]) for each input/output variable, the fuzzy rule [IF X_1 is S and X_2 is {M or L} THEN Y is M] is encoded as [100|011||2]. A chromosome will be the concatenation of a number of these fuzzy rules, e.g., [100|011||2 010|111||1 001|101||3] for a set of three rules.

It is allowed a variable with all the labels set to '1' (which means the variable is not considered it the corresponding rule), but it is forbidden a variable with all the labels set to '0'. It is so because, although one could think on assigning this latter combination to the fact of not using the variable (as in the case of all the labels set to '1'), then we would have solutions genotypically closer but phenotypically far, which is not recommended.

B. Initialization

Since we are looking for optimal completeness, we need starting with rules which cover all the examples. Because of that, we use the well-know Wang-Mendel algorithm [2] to generate the antecedent structure. Specifically, every chromosome is generated with the minimum number of rules that cover the examples according to this algorithm and with a random consequent for every rule (except one chromosome that use the consequents provided by Wang-Mendel). In this way, all chromosomes start with the same number of rules, being so specific as it is possible (i.e., with Mamdani structure instead the DNF one).

C. Covering Hypermatrix Computation

The objective of this step is to generate a data structure which will be used when generating new rules to avoid over-generality. This structure, that we have called *covering hypermatrix*, stores the label combinations of the antecedent that cover everyone of the examples in the data set. Notice that the hypermatrix represents the highest allowed input covering, but it does not show whether a lower number of rules would completely cover the training data set or not, so it can not be used to ensure completeness.

The structure of this hypermatrix is an array which dimension is equal to the number of input variables, containing '1' in a cell if the corresponding input combination cover at least an (n-dimensional) example and containing '0' in other case. With this structure it is possible to design an efficient mutation operator to avoid over-general rules.

The implementation of this structure must be specially efficient, because of its high requirements of access time to the information. In this work we decided implement the hypermatrix using a hash table, which keys are built with the label concatenation of the contained rules. In order to optimize the table in space and information retrieve time, the combinations '0' are not stored. We consider that if a particular key does not exist then its value is '0'.

D. Crossover Operator

The crossover operator only interchanges rules between the two parents, but it does not modify them. Furthermore, it guarantees the children does not present neither inconsistencies nor redundancies. The pseudo-code can be seen in Figure 1.

Function: Crossover Operator.
Input: Two individuals (parents).
Output: Two new individuals (children).
Preconditions: The received individuals have not internal inconsistencies.
Postconditions: The generated individuals have not neither inconsistencies nor redundancies by submission, but it is possible redundancies by partial overlapping. Lack of completeness can also appear.

1) Put all the rules of the parents into a set, S.
2) Analyze those inconsistent rules among them (which always will come from two parents, due to the preconditions). These rules do not have to be inconsistent in pairs. For instance, a rule of the first parent can be inconsistent with two rules of the second one.
3) Divide every group of inconsistent rules into two subsets depending on the parent of which come from and take these rules out S. Assign each subset to a different children at random.
4) Take a random number $r \sim U[1, |S|]$, which will give the number of rules that will be assigned to the first child, being the rest $(|S| - r)$ the number of rules assigned to the second child.
5) Choose at random r rules from S and assign them to the first child.
6) Assign the rest to the second child.
7) This process can generate redundancies in the children. To avoid it, for every child, check if the antecedent of every rule is subsummed by another one and, if so, delete the more specific rules.

Fig. 1. Crossover operator

E. Antecedent Mutation Operator

This operator together with the consequent mutation are the ones that create new rules. As its name suggests, it acts on antecedent variables. When a gene in the antecedent part of a fuzzy rule is chosen to be mutated, the operator analyzes among the available movements (it will be explained below) those that ensures to keep consistency and non-overgenerality (this later case is quickly checked with the covering hypermatrix). The consistency is checked by analyzing the collision of the candidate mutate rule with the rest of them. An option among the available ones is randomly chosen. Therefore, the antecedent mutation operator only explores feasible solutions, thus constraining the search space and ensuring a better exploration.

Figure 2 shows the pseudo-code of the operator. The three different actions are explained in the following.

1) Contraction operation: It converts the mutated rule into a more specific one by choosing a gene of the selected variable with a '1' and flipping to '0'. Clearly, the contraction operator can only be applied when there are, at least, two '1', because in other case all the genes of this variable will be '0' and, as mentioned in Sect. III-A, it is not allowed.

This operator will never cause inconsistency, redundancy or over-generality since it generates a more specific rule, thus avoiding to go into conflict with other rules. The only undesired property it could cause is lack of completeness, but it will be solved by the completeness operator later.

2) Expansion operation: This operator carries out the opposite process to contraction operator, making the rule to

Function: Antecedent Mutation Operator.
Input: One individual and the covering hypermatrix.
Output: The input individual mutated in its antecedent.
Preconditions: The received individual has not internal inconsistencies.
Postconditions: The generated individual has not neither inconsistencies nor subsumed rules, but it is possible redundancies by partial overlapping. Lack of completeness can also appear.

1) Randomly choose an specific input variable to be mutated.
2) Choose at random among some of the following operations: contraction, expansion or shift. Sometimes, it will not be possible to apply some of these operations. The selection of one of these mechanisms is made randomly among the available choices. Notice that there will be always possible to apply at least one movement.

Fig. 2. Antecedent Mutation Operator

Function: Completeness Operator.
Input: One individual and the training data set.
Output: The input individual with some fuzzy rules added, if needed.
Preconditions: None.
Postconditions: The generated individual covers the whole data set, i.e., at least a fuzzy rule matches each example.

1) From the data set, extract such examples that are not covered by any fuzzy rule encoded in the analyzed individual.
2) Apply Wang-Mendel Algorithm over this example subset to generate the minimum number of Mamdani-type fuzzy rules that represent them.
3) Add these fuzzy rules to the rule base encoded in the individual. Since these rules come from data that were not covered by any previous rule, neither consistency nor redundancy problems arise.

Fig. 3. Completeness Operator

be more general. It chooses a gene with allele '0' and flip it to '1'. In this case, the restriction is that the expansion operation can only be applied when there is, at least, a '0' in the genes of the variable.

Unfortunately, this operator can cause collision problems with other rules or generate over-general rules. Therefore, firstly the set of expansion movements that can be applied to the selected variable without causing inconsistencies or over-generality (this latter case is checked using the covering hypermatrix) are generated, and then one of them is randomly chosen. If after performing expansion the mutated rule subsumes other rules, the more specific ones are removed. With this operation it is not possible to get lack of completeness.

3) Shift operation: In this case, the rule does not turn into more general or more specific rule but rather it is moved to a neighboring input subspace. The operation mode of this operator is simple: flip to '0' a gene of the variable and flip to '1' the gene immediately before or after it. If the gene is the first or last label of the variable, only one movement is allowed.

As in the expansion operation, it is possible to get collision problems. The same procedure will be followed to avoid it. Therefore, firstly it is analyzed the valid movements that do not cause inconsistencies or over-generality (checked with the hypermatrix) and one of them are chosen at random. If redundancy appears, the subsumed rules are removed. As in the contraction case, lack of completeness can appear; again it will be solved by the completeness operator.

F. Consequent Mutation Operator

This operator creates new rules by changing the consequent. It simply consists on randomly selecting an output variable of a rule that is not partially overlapped with other rules (it would be the only problematic case since the consequent mutation operator receives consistent and non-subsumed rules). Then, the consequent is randomly changed to the immediately higher or lower linguistic term. The operation does not cause over-generality or lack of completeness.

G. Completeness Operator

The crossover operator and the antecedent mutation by contraction or shift can produce fuzzy rule bases that do not cover some specific data set examples. It is fixed with this operator by adding rules to patch the uncovered input subspaces. In can be considered a reparation operator with a low incidence since it does not change the generated rules, it only add new ones. Figure 3 shows its operation mode.

H. Inference Mechanism

When using DNF-type fuzzy rules, special care must be taken on the inference engine. Indeed, for a proper behavior of the algorithm, it is mandatory to ensure that given two linguistically equivalent rule bases, they are also numerically equivalent. In order to do so, we consider FATI (first aggregate, then inference) approach, the Max-Min scheme (i.e., T-conorm of maximum as aggregation and T-norm of minimum as implication operator), and the T-conorm of maximum as disjunction. Apart from that, T-norm of minimum as conjunction and center-of-gravity as defuzzification are used. These two latter operators could be changed without incurring in a linguistic-numeric discrepancy.

I. Selection and Objective Functions

A generational approach with the multiobjective NSGA-II replacement strategy [18] is considered. Binary tournament selection based on the crowding distance in the objective function space is used.

We consider two objective functions to assess the quality of the generated fuzzy systems, the former (approximation error) to improve the accuracy and the latter (complexity) to improve the interpretability.

- *Aproximation error*: The mean square error (MSE) is used. It is computed as follows:

$$F_1(S) = \frac{1}{N}\sum_{i=1}^{N}(S(x^i) - y^i)^2, \quad (1)$$

with S being the fuzzy system to be evaluated, N the data set size and (x^i, y^i) the ith input-output pair of the data set. The objective is to minimize this function.
- *Complexity*: As complexity measure, we simply use the number of DNF-type fuzzy rules:

$$F_2(S) = |S|. \quad (2)$$

The objective is to minimize this function.
Since the algorithm is designed to ensure optimal covering, i.e., without lack of completeness or over-generalization, we do not care on the linguistic complexity (i.e., generalization) of each fuzzy rule. In a natural way, the most general (i.e., with more labels considered in each rule) the fuzzy rules, the fewer the number of rules. It is an advantage of our approach that simplify the design of this second objective function.

IV. Experimental Results

This section includes the obtained results of the proposed algorithm in two real-world modeling problems with both input and output taking real values:

- The former problem (ELE1) relates to estimation of the low voltage electrical line length in rural towns. It consists of 2 input and one output variables, with a total of 495 examples. Seven uniformly distributed triangular membership functions are used for each input and output variable.
- The latter problem (ELE2) concerns the estimation of electrical network maintenance costs. It consists of 4 input and one output variables and a total of 1056 examples. Five uniformly distributed triangular membership functions are used for each input and output variable.

The experiments shown in this paper have been performed with a 5-fold cross validation. Thus, the data set is divided into five subsets of (approximately) equal size. The algorithm is then applied five times to each problem, each time leaving out one of the subsets from training, but using only the omitted subset to compute the test error. The training and test data partitions used in these two problems are freely available at `http://decsai.ugr.es/~casillas/FMLib/`.

We have considered several learning methods for comparision:

- *Wang and Mendel* [2]: It is a simple algorithm that, in spite of do not obtaining accurate results, is a traditional reference in the research community.
- *Thrift* [7]: It is a classic Pittsburgh-style GA-based fuzzy rule learning method.
- *COR-BWAS* [19]: It is an ant colony optimization based learning algorithm with a great performance between interpretability and accuracy. We have avoided fuzzy rule selection since the algorithm does not guarantee total completeness and the results could not be directly compared with our proposal.
- *Pittsburgh* [20]: It is a Pittsburgh-style GA that also learns DNF-type fuzzy rules. A generational approach and direct replacement are performed, with elitism of the best solution. The fitness is the MSE (eq. 1). The pool is randomly initialized and binary tournament selection is done. The same length-variable coding scheme used in this paper is considered.

Our algorithm has been run with the following parameter values: 300 iterations, 60 population size, 0.7 as crossover probability, and 0.2 as antecedent and consequent mutation probability per chromosome.

Table I and II collects the obtained results, where #R stands for the number of fuzzy rules and MSE_{tra} and MSE_{tst} the approximation error (eq. 1) values over the training and test data set, respectively, rounded to the closer integer value. In our multiobjective algorithm, these values corresponds to the most accurate solution of the final Pareto-optimal set. $\bar{x}$ represents the arithmetic mean of each value over the five partitions and σ the corresponding standard deviation. The best mean results for each problem are shown in boldface.

TABLE I
Results obtained in ELE1 problem

	#R		MSE_{tra}		MSE_{tst}	
Method	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ
Wang-Mendel	22.0	1.4	423 466	8 069	455 262	19 943
Thrift	48.3	0.8	**333 062**	2 804	419 408	34 806
COR-BWAS	22.0	1.4	354 304	7 065	417 142	9 823
Pittsburgh	15.4	3.4	374 595	22 114	460 404	67 597
Pitts-DNF	**13.8**	1.8	336 625	2 270	**414 026**	40 448

TABLE II
Results obtained in ELE2 problem

	#R		MSE_{tra}		MSE_{tst}	
Method	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ
Wang-Mendel	65.0	0.0	112 270	1 498	112 718	4 685
Thrift	565.3	1.8	62 456	1 108	75 158	7 279
COR-BWAS	65.0	0.0	102 664	1 080	102 740	4 321
Pittsburgh	270.0	29.9	174 399	22 237	238 413	47 063
Pitts-DNF	**52.6**	3.1	**62 156**	3 875	**69 268**	9 429

From the obtained results we can observe that the proposed method obtain fuzzy models with a good degree of accuracy and interpretability. It obtains, moreover, the most accurate results with the exception of the training error in the ELE1 problem, where it is only 1% worst than Thrift. Nevertheless, a significantly lower number of rules (71%) are generated in our method. In the ELE2 problem, our method obtains 90% of less rules than the second most accurate solution.

Compared with the other Pittsburgh approach with DNF-type fuzzy rules, the results obtained in ELE2 shows how the constraints of the search space imposed by Pitts-DNF dramatically improve the search process, being significantly more accurate and interpretable than the other algorithm.

This leads us to think the algorithm properly deals the curse of dimensionality.

Furthermore, Figures 4 and 5 show the joint Pareto front obtained by the proposed Pitts-DNF algorithm in both problems. As it can be appreciated, the algorithm generates a large range of solutions with different interpretability-accuracy balances.

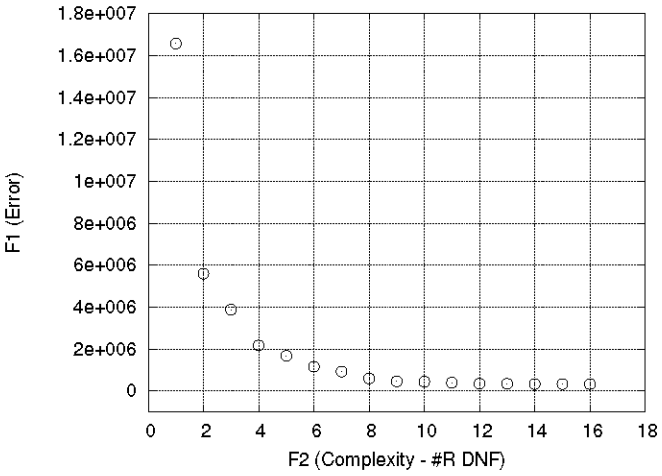

Fig. 4. Pareto front obtained in the ELE1 problem

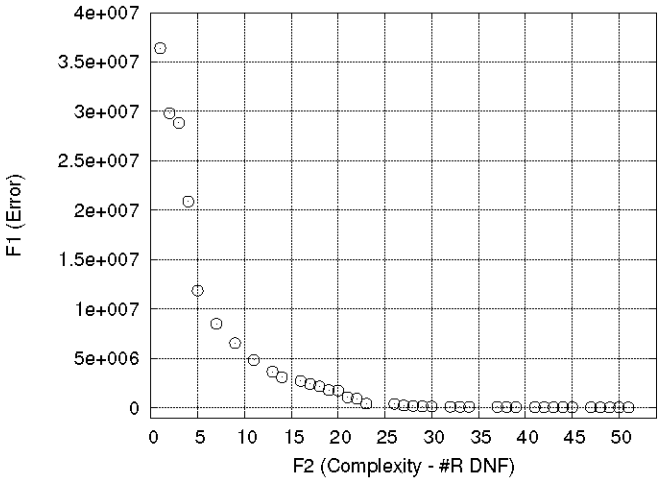

Fig. 5. Pareto front obtained in the ELE2 problem

V. Conclusions

The paper has proposed a genetic fuzzy system for predictive induction in modeling that reaches a high degree of interpretability and accuracy. To do that, the membership functions are keep invariable (thus avoiding losing interpretability). The use of fuzzy rules with conjunctive normal form as antecedent is proposed for a compact knowledge representation capability. This kind of fuzzy rules presents a several problems, such as lack of completeness, inconsistency, redundancy, or over-generality, when the whole rule set is learn at the same time. In this paper, specific genetic operators are proposed to deal with that and its behavior is tested on two real-world modeling problems. As further work, we intend to adapt the algorithm to classification problems (where the output is a class instead a real value) and perform a deeper analysis of its performance.

Acknowledgment

This work was supported in part by the Spanish Ministry of Education and Science under grant no. TIN2005-08386-C05-01.

References

[1] N. Lavrac, B. Cestnik, D. Gamberger, and P. Flach, "Decision support through subgroup discovery: three case studies and the lessons learned," *Machine Learning*, vol. 57, no. 1-2, pp. 115–143, 2004.

[2] L.-X. Wang and J.M. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 22, no. 6, pp. 1414–1427, 1992.

[3] K. Nozaki, H. Ishibuchi, and H. Tanaka, "A simple but powerful heuristic method for generating fuzzy rules from numerical data," *Fuzzy Sets and Systems*, vol. 86, no. 3, pp. 251–270, 1997.

[4] D. Nauck, F. Klawonn, and R. Kruse, *Foundations of neuro-fuzzy systems*, John Wiley & Sons, New York, NY, USA, 1997.

[5] R. Fullér, *Introduction to neuro-fuzzy systems*, Physica-Verlag, Heidelberg, Germany, 2000.

[6] O. Cordón, F. Herrera, F. Hoffmann, and L. Magdalena, *Genetic fuzzy systems: evolutionary tuning and learning of fuzzy knowledge bases*, World Scientific, Singapore, 2001.

[7] P. Thrift, "Fuzzy logic synthesis with genetic algorithms," in *Proc. 4th Int. Conf. on Genetic Algorithms*, R.K. Belew and L.B. Booker, Eds., San Mateo, CA, USA, 1991, pp. 509–513, Morgan Kaufmann Publishers.

[8] C.L. Karr, "Genetic algorithms for fuzzy controllers," *AI Expert*, vol. 6, no. 2, pp. 26–33, 1991.

[9] M. Valenzuela-Rendón, "The fuzzy classifier system: a classifier system for continuously varying variables," in *Proc. 4th Int. Conf. on Genetic Algorithms*, San Mateo, CA, USA, 1991, pp. 346–353, Morgan Kaufmann Publishers.

[10] J. Casillas, O. Cordón, F. Herrera, and L. Magdalena, Eds., *Interpretability issues in fuzzy modeling*, Springer, Heidelberg, Germany, 2003.

[11] J. Casillas, O. Cordón, F. Herrera, and L. Magdalena, Eds., *Accuracy improvements in linguistic fuzzy modeling*, Springer, Heidelberg, Germany, 2003.

[12] Y. Jin, W. von Seelen, and B. Sendhoff, "On generating FC^3 fuzzy rule systems from data using evolution strategies," *IEEE Trans. Systems, Man, and Cybernetics—Part B: Cybernetics*, vol. 29, no. 4, pp. 829–845, 1999.

[13] J.L. Castro, J.J. Castro-Schez, and J.M. Zurita, "Learning maximal structure rules in fuzzy logic for knowledge acquisition in expert systems," *Fuzzy Sets and Systems*, vol. 101, no. 3, pp. 331–342, 1999.

[14] A. González and R. Pérez, "Completeness and consistency conditions for learning fuzzy rules," *Fuzzy Sets and Systems*, vol. 96, no. 1, pp. 37–51, 1998.

[15] A. González and R. Pérez, "SLAVE: a genetic learning system based on an iterative approach," *IEEE Trans. Fuzzy Systems*, vol. 7, no. 2, pp. 176–191, 1999.

[16] L. Magdalena, "Adapting the gain of an FLC with genetic algorithms," *Int. J. Approximate Reasoning*, vol. 17, no. 4, pp. 327–349, 1997.

[17] N. Xiong and L. Litz, "Fuzzy modeling based on premise optimization," in *Proc. 9th IEEE Int. Conf. on Fuzzy Systems*, San Antonio, TX, USA, 2000, pp. 859–864.

[18] K. Deb, A. Pratap, S. Agarwal, and T. Meyarevian, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[19] J. Casillas, O. Cordón, I. Fernández de Viana, and F. Herrera, "Learning cooperative linguistic fuzzy rules using the best-worst ant system algorithm," *Int. J. Intelligent Systems*, vol. 20, pp. 433–452, 2005.

[20] J. Casillas, O. Delgado, and F.J. Martínez-López, "Predictive knowledge discovery by multiobjective genetic fuzzy systems for estimating consumer behavior models," in *Proc. 4th Int. Conf. in Fuzzy Logic and Technology*, Barcelona, Spain, 2005, pp. 272–278.