Fuzzy-UCS: Preliminary Results

Albert Orriols-Puig[1], Jorge Casillas[2], Ester Bernadó-Mansilla[1]

[1]Group of Research in Intelligent Systems, Enginyeria i Arquitectura La Salle
Ramon Llull University, 08022, Barcelona, Spain
[2]Dept. Computer Science and Artificial Intelligence.
University of Granada, 18071, Granada, Spain

aorriols@salle.url.edu, casillas@decsai.urg.es, esterb@salle.url.edu

ABSTRACT

This paper presents Fuzzy-UCS, a Michigan-style Learning Fuzzy-Classifier System designed for supervised learning tasks. Fuzzy-UCS combines the generalization capabilities of UCS with the good interpretability of fuzzy rules to evolve highly accurate and understandable rulesets. Fuzzy-UCS is tested on a set of real-world problems, and compared to UCS and two of the most used machine learning techniques: C4.5 and SMO. The results show that Fuzzy-UCS is highly competitive to the three learners in terms of performance, and that the fuzzy representation permits a much better understandability of the evolved knowledge. These promising results allow for further investigation on Fuzzy-UCS.

Categories and Subject Descriptors

I.2.6 [**Learning**]: concept learning, knowledge acquisition

General Terms

Algorithms

Keywords

Evolutionary Computation, Genetic Algorithms, Machine Learning, Learning Classifier Systems, Fuzzy Logic

1. INTRODUCTION

Michigan-style Learning Classifier Systems (LCSs) are online machine learning techniques that use Genetic Algorithms (GAs) to evolve a rule-based knowledge. Among the different uses, several LCSs have been designed to be applied on supervised learning problems. Typically, LCSs deal with numerical attributes by means of evolving a set of intervalar rules that cooperate to predict the output of new unlabeled examples. Although the competence of LCSs in terms of accuracy have been widely shown, this excellence has been hindered by a poor interpretability of the evolved rulesets, which typically consist of large sets of overlapping intervalar rules that can hardly be read by the human expert.

During the last decade, the interest in Fuzzy Rule-Based Systems (FRBSs) has increased since they provide a robust, flexible, and powerful methodology to deal with *noisy*, *imprecise* and *incomplete* data. Besides, the fuzzy representation allows for a better interpretability of the rules. This led to the first analyses and designs of *Learning Fuzzy-Classifier Systems* (LFCS) [4], mostly applied to reinforcement learning and control tasks, which resulted in the creation of efficient systems with better linguistic interpretability. These first successful steps toward the design of competent LFCS warrants for further investigation, specially in the supervised learning paradigm.

In this paper, we address the problem of interpretability in LCSs, and propose Fuzzy-UCS, an online accuracy-based LFCS architecture that works under a supervised learning paradigm. We depart from the UCS classifier system, which has been shown to be highly competitive with respect to some of the most used machine learning techniques [1]. We introduce the fuzzy representation to UCS, and redesign most of its components to permit the system to deal with fuzzy rules. With the inclusion of fuzzy rules, we seek for a better interpretability of the evolved knowledge whilst maintaining the same performance, as well as a significant reduction of the search space.

The remaining of this paper is organized as follows. Section 2 deeply explains the proposed Fuzzy-UCS architecture, specially focusing on the differences from the original UCS. In Sect. 3, we test Fuzzy-UCS on a testbed consisting of eight real-world problems, analyze the evolved fuzzy rules, and compare the system to UCS, C4.5, and SMO. Finally, Sect. 4 concludes and proposes further work.

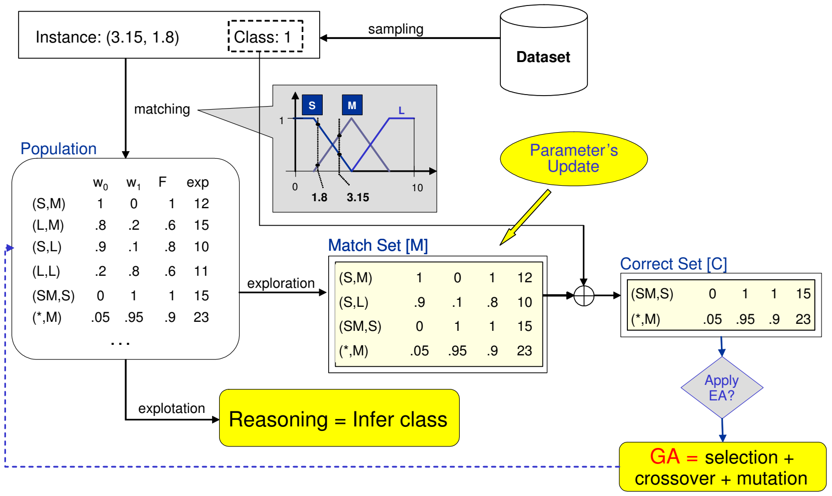
2. DESCRIPTION OF FUZZY-UCS

Figure 1 shows the learning process of Fuzzy-UCS. The learner works in two different modes: learning or exploration mode and testing or exploitation mode. During explore, Fuzzy-UCS evaluates online the classifiers and evolves the ruleset by means of the GA. During test, Fuzzy-UCS uses the rules to infer the output of a given input instance. The different components of the system are detailed as follows.

2.1 Representation

Fuzzy-UCS evolves a population [P] of classifiers, where each classifier consists of a *fuzzy rule* and a set of parameters. The fuzzy rule is represented as follows:

GECCO'07, July 7–11, 2007, London, England, United Kingdom.
Copyright 2007 ACM 978-1-59593-698-1/07/0007 ...$5.00.



Population

	w_0	w_1	F	exp
(S,M)	1	0	1	12
(L,M)	.8	.2	.6	15
(S,L)	.9	.1	.8	10
(L,L)	.2	.8	.6	11
(SM,S)	0	1	1	15
(*,M)	.05	.95	.9	23
. . .				

Match Set [M]

(S,M)	1	0	1	12
(S,L)	.9	.1	.8	10
(SM,S)	0	1	1	15
(*,M)	.05	.95	.9	23

Correct Set [C]

(SM,S)	0	1	1	15
(*,M)	.05	.95	.9	23

Figure 1: Scheme of Fuzzy-UCS.

$$\textbf{IF } x_1 \text{ is } A_1^k \text{ and ... and } x_n \text{ is } A_n^k \textbf{ THEN} \\ c_1^k \textbf{ WITH } r_1^k, \text{ ... }, c_m^k \textbf{ WITH } r_m^k \qquad (1)$$

where each input variable x_i is represented by a disjunction (T-conorm operator) of linguistic terms $A_i^k = \{A_{i1}$ or ... or $A_{in_i}\}$, and the consequent maintains, for each of the m possible classes, a weight r_j that indicates the soundness with which the rule predicts the class j for an example that fully matches this rule. Note that this type of rule intrinsically permits generalization as each variable can take an arbitrary number of linguistic terms. For example, the representation supports the absence of a variable x_i by making A_i be the whole set of linguistic terms.

Each classifier has four main parameters: a) the fitness F, which estimates the accuracy of the rule, b) the correct set size cs, which averages the sizes of the correct sets in which the classifier has participated, c) the experience exp, which reckons the contributions of the rule to classify the input instances, and d) the numerosity num, which counts the number of copies of the classifier in the population.

To implement this representation, we propose to use a binary coding for the antecedent of the rule. That is, a one-valued allele indicates that the corresponding linguistic term is used in this variable. The consequent weights r_j are codified in an array of floats. For instance, if we have three linguistic labels {S [small], M [medium], L [large]} for each input and two possible classes, the fuzzy rule

$$\textbf{IF } x_1 \text{ is } S \text{ and } x_2 \text{ is } \{S \text{ or } L\} \textbf{ THEN } c_1 \\ \textbf{WITH } 0.8, c_2 \textbf{ WITH } 0.2 \qquad (2)$$

is encoded as: [100|101||0.8|0.2].

2.2 Performance Component

UCS learns under a supervised learning scheme. Given an input example e with its associated class c UCS creates the *match set* [M] with all the classifiers in [P] that *match* the input instance. Then, [M] is used differently depending on whether the system is running on explore or on exploit mode. In explore mode, UCS forms the *correct set* [C], which consists of all the classifiers in [M] that advocate the input example. If [C] is empty, *covering* is triggered. In exploit mode, the best action selected from the vote (weighted by fitness) of all classifiers in [M] is returned as the predicted output.

Fuzzy-UCS follows this process, but the role of the matching and the inference processes changes as it is adapted to deal with linguistic terms. In the following, the phases followed during explore are detailed. The inference mechanism used during test is explained in Sect. 2.5.1.

Creation of the match set. Given the input e, all the classifiers with a *matching degree* greater than zero form the match set. The matching degree $\mu_k(e)$ of the rule k is determined as follows. For each variable, we compute the membership function for each of its linguistic terms, and aggregate them by means of a T-conorm. Then, the matching degree of the rule is determined by the T-norm of all the input variables. We used the *product* $(a \cdot b)$ as T-norm and the *bounded sum* $(min\{1, a+b\})$ as T-conorm. Note that we used a bounded sum instead of other typical operators for the T-conorm to emulate the *don't care* used in the crisp representation. That is, with the bounded sum, a variable will have no influence if all its linguistic terms are set to '1'.

Creation of the correct set. Next, all the classifiers in [M] in which their higher weight r_c^k corresponds to the class c of the input example form the correct set[C]. If there is not any rule in [C] that matches e with the maximum matching degree, the covering operator is triggered. We create the rule that matches e with maximum degree as follows. For each variable x_i, the linguistic term with maximum matching with e_i is activated. Then, the rule is generalized by setting each linguistic term to '1' with probability $P_\#$. In the consequent, the weight corresponding to the class c is set to 1, and the others to 0. The parameters of the classifier are initialized to: f=1, exp=0, num=1, and cs is set to the size of [C]. Finally, this rule is introduced in the population if there is not any rule in [C] with the same matching degree.

2.3 Parameter's Update

At the end of each learning iteration, the parameters of all the classifiers that belong to the match set are updated according to their matching degree with the input example e of class c. First, for each class j, we compute the sum of correct matchings cm_j of each classifier k:

$$cm_j^k = cm_j^k + m(k,j) \tag{3}$$

where

$$m(k,j) = \begin{cases} \mu_k(e) & \text{if } j = c \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

that is, we compute separately the sum of matching degrees of every rule with the examples of different classes. We also calculate the sum of all matching degrees sm:

$$sm^k = sm^k + \mu_k(e) \tag{5}$$

Then, the weight of each class is computed as:

$$r_j^k = \frac{cm_j^k}{sm^k} \tag{6}$$

For example, if a rule k only matches examples of class j, the weight r_j^k will be 1 and the remaining weights 0. Rules that match instances of both classes will have weights ranging from 0 to 1. In all cases, the sum of all the weights is 1.

Then, the fitness is computed from the class weights with the aim of favoring classifiers that match instances of only one class. For this purpose, we compute the fitness as follows:

$$F^k = \left(r_{max}^k - \sum_{j|j \neq max} r_j^k \right)^{\nu} \tag{7}$$

where $\nu \geq 0$ permits to regulate the fitness pressure toward accurate classifiers. In the equation, we select the weight r_{max}^k with maximum value and substract the values of the other weights. Note that this formula can result in classifiers with zero or negative fitness (for example, if the class weights are equal).

Next, the correct set size cs^k is calculated as the arithmetic average of all the correct sets the classifier has belonged to. Finally, the experience is computed as the sum of all the matching degrees of the classifier obtained when participating in [M].

2.4 Discovery Component

Similarly to UCS, Fuzzy-UCS uses a genetic algorithm (GA) for discovering new promising rules. The GA is applied on [C] if the average time since its last application on the classifiers in [C] is greater than θ_{GA}. In this case, two parents are selected from [C] with probability proportional to their fitness. The fitness of young classifiers is decreased since they receive a minimum number of updates. Then, the parents are crossed and mutated with probabilities χ and μ respectively. The consequent part and the parameters of the offspring are initialized as in covering.

The crossover operator crosses the rule antecedent by two points. This could result in classifiers containing variables with no linguistic terms, which would indicate that the rule is not applicable. If this is the case, we copy a linguistic term from one of the parents. Crossover does not modify the consequent of the rule.

The mutation operator randomly decides if an attribute has to be mutated. If a variable is selected, three types of mutation can be applied: *expansion*, *contraction*, or *shift*. Expansion chooses a linguistic term not represented in the corresponding variable and adds it to this variable; thus, it can only be applied on variables that do not have all the linguistic terms. Contraction is the opposite process: is removes a linguistic term from one variable; so, it can only be applied if the variable has more than one linguistic term. Shift changes a linguistic term for its immediately inferior or superior.

Finally, the new offspring are inserted into the population. First, each offspring is checked for subsumption with its parents. If either of the parents is enough experienced ($exp > \theta_{sub}$), highly accurate ($F > F_0$), and more general than the offspring, its numerosity is incremented. If the offspring cannot be subsumed by any of its parents, the same process is used to find a subsumer in [C]. If there is no subsumer, the offspring is inserted in the population. A classifier is deleted if the population is full; in this case, each classifier is given a deletion probability of

$$P_{del_k} = \begin{cases} cs_k \cdot num_k \cdot \frac{\overline{F}}{F_k} & \text{if } exp > \theta_{del} \text{ and } \delta\overline{F} > F_k \\ cs_k \cdot num_k & \text{otherwise} \end{cases} \tag{8}$$

where $\overline{F}$ is the average fitness of classifiers in [P], and δ and θ_{del} are two parameters set by the user ($0 < \delta < 1$ and θ_{del} >0). Thus, this method gives a higher deletion probabilities to numerous classifiers that belong to large correct sets; moreover, it also penalizes experienced classifiers with low fitness.

2.5 Fuzzy-UCS in test mode

Fuzzy-UCS aims at obtaining a highly accurate ruleset with minimum size. To obtain high accuracy, we need to define an effective reasoning method to infer the output class from the final population. To obtain a compact ruleset, some reduction strategies may be applied to remove classifiers that are not important for the reasoning. In the following, we discuss two reasoning approaches which lead to two different ruleset reduction mechanisms.

2.5.1 Class Inference

In test mode, given a new unlabeled instance e, several rules can match (with different degrees) this instance, advocating with a certain soundness r_i to each of the classes. Thus, a reasoning process needs to be applied to decide the output. Here, we propose two fuzzy-inference approaches:

Winner rule. This approach proposes to select the rule k that maximizes $\mu_k(e) \cdot F^k$, and chose as output the class associated to the weight with maximum value. Note that following this strategy, the rules can be rewritten as:

$$\textbf{IF } x_1 \text{ is } A_1^k \text{ and... and } x_n \text{ is } A_n^k \textbf{ THEN } c_j \textbf{ WITH } F^k \tag{9}$$

That is, the rule predicts the class j that have a maximum weight w_j with a soundness equal to its fitness, providing a better interpretability.

Average vote. This approach infers the class from the information provided by all the rules with enough experience ($exp > \theta_{exploit}$). Each experienced rule k votes for each action j according to $\mu_k(e) \cdot w_j^k$. We add all the votes per class, and the most-voted class is returned as the output.

Table 1: Comparison of the test accuracy of Fuzzy-UCS with the winner rule inference (WInf) and the average vote inference (AVote), C4.5, and SMO on eight real-world problems. The symbols ○ and • indicate that Fuzzy-UCS with WInf and AVote respectively performed significantly different than the algorithm in the column according to a paired t-test at 0.95 significance level.

	Fuzzy-UCS		UCS	C4.5	SMO
	WRule	Avote			
bal	83.20 •	86.38 ○	77.88 ○•	77.42 ○•	86.89 ○
bpa	65.21	64.08	68.42	62.31	58.28 ○•
irs	96.67	90.00	93.33	94.00	96.67 •
h-c	80.48	83.12	79.15 •	78.45 •	85.15 ○
pim	76.69	75.39	75.39	74.23	76.97
thy	89.33	87.98	84.89	94.91 ○•	88.91
wbcd	96.71	97.57	96.14	94.99	96.85
wne	94.41	97.75	94.97 •	93.89 •	99.44

2.5.2 Ruleset Reduction

At the end of the learning, the population is reduced to obtain a minimum set of rules with the same train accuracy. The applied reduction strategy depends on the type of inference used.

Winner rule. For each instance in the training dataset, we create the match set and infer the output. The classifier used for the inference is copied to the final population. The process is repeated for all the input instances, obtaining a reduced set of rules that achieve the same training accuracy than the whole population.

Average vote. In this strategy, as all the rules participate in the inference process, we can hardly find an efficient process to reduce the population without loss of training accuracy. Thus, in this case, we only remove from the population the rules that are not experienced enough ($exp < \theta_{exploit}$) and those with zero or negative fitness.

3. EXPERIMENTATION

We selected eight real-world problems from the UCI repository [2]: *balance-scale* (bal), *bupa* (bpa), *Heart-C* (h-c), *Iris* (irs), *pima* (pim), *thyroid* (thy), *Wisconsin breast cancer* (wbcd), and *wine* (wne). We ran Fuzzy-UCS on these problems with the two types of inference presented in Sect. 2.5.1 and the following configuration: learning iterations = 100 000, population size 6400, F_0=0.99, ν=10, θ_{GA}=50, χ=0.8, μ=0.04, θ_{del}=50, θ_{sub}=50, δ=0.1, $P_{\#}$=0.6. Moreover, we used five triangular shape linguistic terms uniformly distributed. We compared the results to those obtained by UCS, C4.5, and SMO with lineal kernels (C4.5 and SMO were run using WEKA [5]). To get good estimates of the test accuracy, we used a ten-fold cross validation [3].

Table 1 summarizes the results obtained with the five learners. Fuzzy-UCS with both inference systems present highly competitive results with respect to the other learners. Note that Fuzzy-UCS presents similar test accuracies than those of UCS, and that outperforms C4.5, one of the most-known and widely-used machine learning technique, in most of the domains.

Besides its high accuracy, Fuzzy-UCS creates models that are easier to understand than those built by the other learners. For example, let us compare the interpretability of rule-based knowledge evolved by UCS and Fuzzy-UCS in the *irs* problem. UCS evolved rulesets that consisted of 800 rules on average, in which the most numerous and accurate rules created for each class were:

1. **IF** $x_1 \in$ [4.30, 7.76] and $x_2 \in$ [2.91, 4.40] and $x_3 \in$ [1.00, 3.77] and $x_4 \in$ [0.10, 2.19] **THEN** Iris-setosa
2. **IF** $x_1 \in$ [4.30, 6.06] and $x_2 \in$ [2.00, 2.53] and $x_3 \in$ [4.30, 6.90] and $x_4 \in$ [0.10, 2.50] **THEN** Iris-virginica
3. **IF** $x_1 \in$ [4.30, 7.71] and $x_2 \in$ [2.00, 3.90] and $x_3 \in$ [2.63, 6.90] and $x_4 \in$ [0.10, 1.39] **THEN** Iris-versicolor

Fuzzy-UCS with average vote evolved a similar number of rules, all of them used in the inference process. On the other hand, Fuzzy-UCS with a winner rule inference evolved an average of 20 rules. Having configured the system with five linguistic terms {XS,S,M,L,XL}, the most numerous and accurate rules evolved for each class were:

1. **IF** x_3 is {XS, S or L} and x_4 is {XS or S} **THEN** Iris-setosa **WITH** F =1
2. **IF** x_1 is not M and x_4 is {L or XL} **THEN** Iris-virginica **WITH** F =1
3. **IF** x_3 is M and x_4 is not L **THEN** Iris-versicolor **WITH** F =1

Note that some of the variables are completely generalized. Moreover, the fuzzy rules generated by Fuzzy-UCS are by far easier to be interpreted than the intervalar rules evolved by UCS.

4. CONCLUSIONS AND FURTHER WORK

This paper proposed Fuzzy-UCS, an LFCS that uses a GA to evolve a set of fuzzy rules. We showed that the performance of Fuzzy-UCS is comparable to those obtained with UCS, C4.5 and SMO, and that the interpretability of the fuzzy rules created is much better than those evolved by UCS. These promising results lead to further investigation. As further work, we will compare Fuzzy-UCS to other fuzzy and crisp machine learning methods on a large set of datasets. Moreover, a deeper analysis on the generalization capacity of Fuzzy-UCS will be conducted, as well as on the interpretability of the fuzzy-rules compared to other knowledge representations.

Acknowledgements

This work was supported by MCyT under projects TIN2005-08386-C05-01 and TIN2005-08386-C05-04, and *Generalitat de Catalunya* under grants 2005FI-00252 and 2005SGR-00302.

5. REFERENCES

[1] E. Bernadó-Mansilla and J. Garrell. Accuracy-Based Learning Classifier Systems: Models, Analysis and Applications to Classification Tasks. *Evolutionary Computation*, 11(3):209–238, 2003.

[2] C. Blake and C. Merz. *UCI Repository of ML Databases: http://www.ics.uc.edu/ mlearn/MLRepository.html*. Univ. of California, 1998.

[3] T.G. Dietterich. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Comp.*, 10(7):1895–1924, 1998.

[4] M. Valenzuela-Radón. The Fuzzy Classifier System: A Classifier System for Continuously Varying Variables. In *4th ICGA*, pages 346–353. Morgan Kaufmann, 1991.

[5] I. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, 2nd edition, 2005.