Fuzzy-UCS: A Michigan-Style Learning Fuzzy-Classifier System for Supervised Learning

Albert Orriols-Puig, Jorge Casillas, and Ester Bernadó-Mansilla

***Abstract*—This paper presents Fuzzy-UCS, a Michigan-style Learning Fuzzy-Classifier System specifically designed for supervised learning tasks. Fuzzy-UCS is inspired by UCS, an on-line accuracy-based Learning Classifier System. Fuzzy-UCS introduces a linguistic representation of the rules with the aim of evolving more readable rule sets, while maintaining similar performance and generalization capabilities to those presented by UCS. The behavior of Fuzzy-UCS is analyzed in detail from several perspectives. The granularity of the linguistic fuzzy representation to define complex decision boundaries is illustrated graphically, and the test performance obtained with different inference schemes is studied. Fuzzy-UCS is also compared with a large set of other fuzzy and nonfuzzy learners, demonstrating the competitiveness of its on-line architecture in terms of performance and interpretability. Finally, the paper shows the advantages obtained when Fuzzy-UCS is applied to learn fuzzy models from large volumes of data.**

***Index Terms*—Genetic fuzzy systems, Michigan-style learning classifier systems, pattern classification, supervised learning.**

I. Introduction

MICHIGAN-STYLE Evolutionary Learning Systems (introduced by Holland in the 1970s [1], [2]), also referred to as Learning Classifier Systems (LCSs), are machine learning techniques that solve problems on-line by evolving a set of rules by means of an Evolutionary Algorithm (EA). Initially designed based on animal behavior, new developments in research on Michigan-style LCSs—mostly due to the presentation of XCS (originally proposed in [3] and further improved in [4]), the first accuracy-based LCS—have led to the application of the on-line learning architecture to solve pattern classification tasks [5], reinforcement learning problems [6], and function approximation problems [7].

In the pattern recognition field, several studies and comparisons demonstrate the competitiveness of the performance of LCSs with respect to other widely-used machine learning techniques [5], [8]–[10], such as the decision tree C4.5 [11], the support vector machine SMO [12], and the instance based algorithm IBk [13]. LCS's success is due to three main factors: the rule-based representation, the generalization capabilities, and the on-line learning scheme. That is to say, starting from a set of rules approximately created from the first input examples, the system estimates the quality of these rules on-line and the genetic search incrementally evolves this rule set with the aim of obtaining a set of maximally general and accurate rules which together cover all the input space. Besides, the population-based architecture of LCSs permits their parallelization for use on supercomputing resources [14], making them also competitive for mining large data sets.

The high competence of LCSs to perform pattern recognition tasks has been impaired to some extent by the large number of semantic-free rules that are evolved. LCSs represent continuous variables by means of interval-based rules; i.e., each rule represents a codification of a hyper rectangle in the feature space which is usually coded by using real numbers. This results in large sets of overlapped rules which together define the decision boundaries [15]. Thus, this knowledge representation is barely legible to human experts. Some authors have tried to solve this problem by designing rule set reduction algorithms [16]–[18]. Although some of these algorithms allow for a considerable reduction of the rule sets, slightly degrading the test performance, the semantic-free descriptive representation may continue hampering the readability of the rule set.

In recent years, there has been an increasing interest in *Genetic Fuzzy Rule-Based Systems* (GFRBSs) [19]—which mainly involve the use of evolutionary algorithms to learn fuzzy rules—since they provide a robust, flexible, and powerful methodology to deal with a highly legible knowledge representation. As a result, the first Michigan-style *Learning Fuzzy-Classifier Systems* (LFCSs) have been proposed [20]–[26], most of them applied to solve reinforcement learning and process control tasks. However, while most of the current nonfuzzy Michigan-style LCSs are accuracy-based, the fuzzy approaches usually follow a strength-based model, in which classifiers are strengthened during the learning process.

In this paper, we address the interpretability problem in LCSs and propose Fuzzy-UCS, an accuracy-based Michigan-style LFCS that works under a supervised learning paradigm. For this purpose, we use UCS [5] as starting point. UCS is an LCS derived from XCS and specialized for pattern recognition tasks, which has shown to be highly competitive with respect to other machine learning techniques. Fuzzy-UCS replaces the interval-based representation with a linguistic representation of the rules, and redefines the majority of UCSs' components to deal with fuzzy rules. The system learns incrementally from

Manuscript received July 31, 2007; revised November 30, 2007 and January 31, 2008. First published August 08, 2008; current version published April 01, 2009. This work was supported by the Ministerio de Educación y Ciencia under Projects TIN2005-08386-C05-01 and TIN2005-08386-C05-04 and by the Generalitat de Catalunya under Grants 2005FI-00252 and 2005SGR-00302.

A. Orriols-Puig and E. Bernadó-Mansilla are with the Grup de Recerca en Sistemes Intel·ligents, Enginyeria i Arquitectura La Salle, Universitat Ramon Llull, 08022 Barcelona, Catalonia, Spain (e-mail: aorriols@salle.url.edu; esterb@salle.url.edu).

J. Casillas is with the Department of Computer Science and Artificial Intelligence, Universidad de Granada, 18071 Granada, Spain (e-mail: casillas@decsai.ugr.es).

Digital Object Identifier 10.1109/TEVC.2008.925144

a stream of examples, which are used to adjust the quality estimate of the fuzzy rules. Also, a steady-state niched *genetic algorithm* [27], [28] is applied periodically to create promising new rules. At the end of the learning, the system provides a set of maximally general and accurate rules which may present overlapping conditions.

The remainder of this paper is organized as follows. Section II briefly presents the Rule-Based Evolutionary Algorithms, discusses the main characteristics of Michigan and Pittsburgh-style LCSs, introduces accuracy-based LCSs (focusing on UCS) and reviews some of the Michigan-style LFCS proposed in the literature. Section III gives a detailed description of the proposed Fuzzy-UCS algorithm. The Section II analyzes the limitations that a linguistic representation may impose, and compares three inference schemes proposed in Fuzzy-UCS. Section V makes an extensive comparison of the Fuzzy-UCS representation with a set of GFRBS and general-purpose machine learning techniques, analyzing the differences between the learners in terms of test performance and interpretability of the models. Section VI exploits the on-line architecture of Fuzzy-UCS to mine a large data set, the 1999 KDD Cup intrusion detection data set. Finally, Section VII gives a summary of the work and future lines of research.

II. Framework: On-Line Accuracy-Based LCSs

This section introduces the different rule-based evolutionary learning models, i.e., machine learning techniques that use Evolutionary Algorithms (EAs) to evolve the rule-based knowledge. We briefly describe the different strategies proposed in the literature, and focus on the accuracy-based Michigan-style LCSs. In this context, we introduce UCS [5], an LCS designed for supervised learning by which Fuzzy-UCS is inspired. Finally, we review the related work on Learning Fuzzy-Classifier Systems.

A. Rule-Based Evolutionary Learning Systems

Since Holland presented the first architecture of Learning Classifier Systems in 1976 [1], later implemented in 1978 [2], research on LCSs has been conducted from two different perspectives: the Pittsburgh-style LCSs [29], and the Michigan-style LCSs [1], [2]. Both types of algorithms are briefly described as follows:

1) *Pittsburgh-style LCSs* follow the essence of *evolutionary algorithms*. Every individual is a set of production rules that represent a solution for the given problem. The quality of each individual is estimated according to the number of examples correctly predicted and the rule's generalization. All the solutions compete in the population. The genetic search is usually driven by a generational genetic algorithm [27], [28], whose operators are adapted to deal with rule sets. At the end of the run, the fittest individual is selected for classifying new test examples.
2) *Michigan-style LCSs* are *cognitive* systems that combine a credit-apportionment algorithm, usually based on reinforcement learning [30], with evolutionary algorithms [27]. Every individual is a single production rule, whose quality is evaluated on-line by the cognitive system. An evolutionary algorithm is applied periodically to the population to discover promising new rules. At the end of the run, all the rules in the population are grouped together to classify new test instances.

The two strategies mainly differ in a) the individual representation, b) the evaluation of the individuals, and c) the application of the EA.

Pittsburgh-style LCSs represent each individual as a set of rules; thus, the genetic operators evolve individuals that classify all the input space. On the other hand, Michigan-style LCSs codify each individual as a single production rule. Consequently, each rule (also referred to as *classifier*), is an expert classifier in the region of the search space that it covers. Thus, since all the evolved classifiers collaborate to cover all the feature space, a methodology for combining classifiers with overlapping conditions is required.

To evaluate an individual, Pittsburgh-style LCSs classify all the input examples in the training data set. Thus, the computational resources needed to evaluate all the rule set increase linearly with the number of instances in the training data set. This is one of the main problems detected for Pittsburgh-style LCSs, especially when used to mine big amounts of data. On the contrary, Michigan-style LCSs evaluate the population on-line by interacting with an environment which provides an example at each learning iteration. Consequently, the number of instances in the data set does not influence the cost of performing a learning iteration.

The application of the evolutionary algorithm also differs in both approaches. Typically, in Pittsburgh-style LCSs, a generational EA is applied at the rule set level: selection, crossover, and mutation are adapted to deal with individuals codified as rule sets. In Michigan-style LCSs, a steady-state EA works at the rule level, since individuals codify a single rule. The consequence of the latter approach is that the EA must co-evolve a diverse set of individuals which together provide a solution to the problem.

Recently, new proposals that hybridize Michigan- and Pittsburgh-style LCSs have been proposed. For example, in [31] a hybrid of both LCSs styles is presented to extract rule sets for classification problems.

B. UCS: An Accuracy-Based Michigan-Style LCS

In recent years, many implementations of Michigan-style LCSs have been proposed. Most of them use a reinforcement learning procedure to evaluate the rule set on-line. These approaches can be grouped in two categories, depending on how they compute the rule's fitness: a) *strength-based* LCSs and b) *accuracy-based* LCSs. The first practical implementations of LCSs corresponded to strength-based LCSs. In these systems, rules' fitness was based on the strength, i.e., an estimate of the reward that the system would receive if the action of the rule was performed. However, several limitations were identified in this approach such as the presence of over-general classifiers, due to the difficulty of distinguishing them from accurate classifiers [32].

During this period of increasing research on LCSs, Wilson presented XCS [3], [4] which came to solve the typical problems of strength-based LCSs with the idea of basing fitness on the accuracy of the reward prediction instead of on the reward itself. This means that the evolutionary algorithm searches

for rules that are accurate in their prediction, regardless of the expected reward of each rule. This new architecture also brought another view of the kind of rules that need to be evolved. Whilst strength-based LCSs only need to maintain all the highly-rewarded rules (i.e., rules that receive a high payoff), accuracy-based LCSs such as XCS need to create all possible classifiers with minimum prediction error, regardless of the payoff they receive.

Since the introduction of XCS, a great amount of research has been conducted on accuracy-based LCSs, resulting in different LCSs with a core architecture inherited from XCS. One of the most prominent proposals is UCS [5], which inherits the main components of XCS, but specifies them for supervised learning tasks. In [5] and [33], UCS was able to overcome the *fitness dilemma* [9] detected in XCS and to achieve accurate models quicker than XCS; moreover, UCS reached higher accuracy rates in imbalanced problems and multi-class problems. The on-line architecture enabled learning from a stream of examples, without going through the whole data set in each learning iteration. This feature is really useful for learning incrementally from large data sets, as shown later in this paper.

UCS works as a model-free on-line learner. For each input example e with its associated output c, UCS forms the *match set* [M], which consists of all the classifiers in the population [P] with their matching condition e. The next steps depend on whether the system is on *exploration* (or training) mode or *exploitation* (or test) mode. Under exploration mode, the system creates the *correct set* [C] with all classifiers in [M] that advocate c. If [C] is empty, the covering operator is triggered. It creates a new rule whose condition is generalized from e and which predicts the class c. Then, the parameters of the all rules in [M] are updated depending on whether they predicted e correctly. Eventually, a genetic algorithm is triggered on the correct set [C], creating two new classifiers by means of crossover and mutation. The offspring are introduced in the population, and other classifiers are removed from the population if there is no room for the new rules. The combination of niched-based selection and population-based replacement is mainly responsible for the generalization pressure in UCS. Under exploitation mode, each classifier in [M] emits a vote weighted by the fitness of the rule for the class it predicts. The most voted class is selected as output.

C. Related Work on Learning Fuzzy-Classifier Systems

Several authors have proposed strength-based Michigan style LFCS, which have basically been applied to solve reinforcement learning and control tasks. Valenzuela-Rendón [20] introduced the first Michigan-style LFCS, which consisted of a fixed-size fuzzy-rule set and a fuzzy message list. The system was applied to solve function approximation tasks. The quality of the fuzzy rules was given according to the accuracy in which the output was estimated. Thus, the initial approach was not a pure reinforcement learning architecture. The system was later enhanced with true reinforcement learning [21].

Several strength-based Michigan-style LFCS have been proposed since [21]. Parodi and Bonelli [22] presented an LFCS that automatically learned fuzzy relations, fuzzy membership functions, and fuzzy weights. The fitness (strength) of each rule was used for a double purpose. First, it served to compute the selection and replacement probability of the rule. Second, it permitted stronger rules to participate more soundly in the inference process.

Furuhashi *et al.* [23] designed an LFCS that used multiple stimulus-response fuzzy rules operating in tandem. The system was applied to a control task in which a simulated ship had to reach a target without moving the obstacles found on its way. The same problem was addressed by Nakaoka *et al.* by using a single rule list [34].

Velasco [24] defined a new LFCS architecture designed for fuzzy process control. The system introduced the so-called limbos, i.e., a special workspace where new rules were generated and evaluated before being used in the real process plant. In this way, the system avoided using poorly-evaluated rules in the control system.

Ishibuchi *et al.* [25] designed one of the first proposals of LFCS for pattern classification. They used a fixed-size rule set where *don't care* symbols were defined to permit generalization in the fuzzy rules. A certainty factor, derived from a heuristic procedure prior to fitness evaluation, together with the predicted class formed the consequent of the rule. An evolutionary algorithm, which operated only on the rule antecedent, was responsible for creating promising new rules. Recently, a hybridization of Pittsburgh-style and Michigan-style LCSs has been presented by two of the aforementioned authors [35], in which a single iteration of a Michigan-style LCS—i.e., rule selection, generation, and replacement— is applied to each individual of a Pittsburgh-style rule set.

Finally, the classic "competition versus cooperation" problem in genetic fuzzy systems was addressed in Bonarini's work [36], [37]. Bonarini proposed a Michigan-style LCS called ELF, which faced the dilemma between the desired cooperation among fuzzy rules that match a given input state and the competition of these rules in the evolutionary algorithm. In ELF, the rule set was divided into several subpopulations, each one with the same antecedent. Then, the rules of different subpopulations cooperated to produce the control action, whilst the members of each subpopulation competed with each other. Moreover, ELF controlled the instability of general rules that participated in different subpopulations by providing each rule a reinforcement normalized on the difference between the maximum and the minimum reinforcement obtained by the subpopulation to which the rule belongs. In this way, ELF overcame some of the problems of strength-based LCSs. ELF was applied to several reinforcement learning problems, such as the coordination of autonomous agents.

All the LFCS described through this section are strength-based systems. In reinforcement learning, the first successful accuracy-based fuzzy rule-based system with generalization capability was proposed in [26]. To the best of our knowledge, no accuracy-based LFCS specifically designed for classification has been proposed.

In our system, we take an accuracy-based approach to benefit from the advantages that these types of systems have introduced to LCSs, which are summarized as follows.

- Accuracy-based LCSs can distinguish over-general from accurate rules [38].

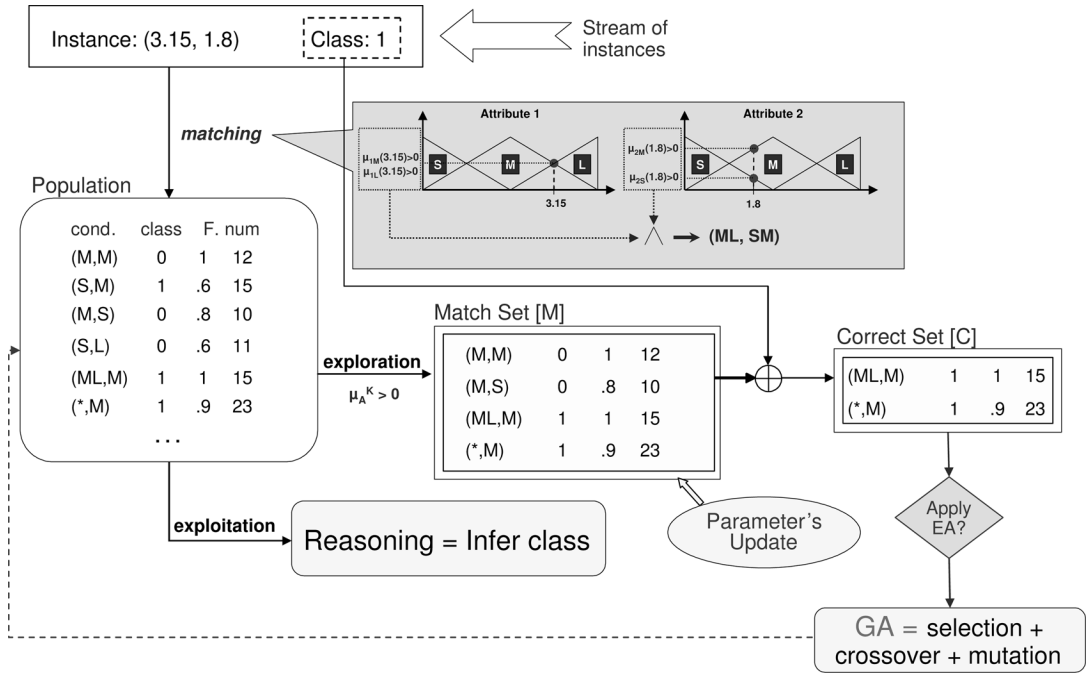


Fig. 1. Schematic illustration of Fuzzy-UCS. The run cycle depends on the type of run: exploration (training) or exploitation (test).

- There are theoretical analyses that support the theory that, for binary representation, LCSs such as XCS will evolve a rule set with maximally-general and highly accurate rules if certain conditions are met [9], [39], [40]. Although similar analyses in the continuous space are lacking, the positive conclusions extracted for the binary representation promote the use of Michigan-style LCSs.

Moreover, we have designed our system to solve classification tasks. For this purpose, our system is inspired by UCS, one of the most significant LCSs for supervised learning. In Section III we introduce the system, which is addressed as Fuzzy-UCS.

III. Description of Fuzzy-UCS

Fig. 1 schematically illustrates Fuzzy-UCS. The system works in two different modes: *exploration* or training and *exploitation* or test. In the exploration mode, Fuzzy-UCS seeks to evolve a maximally general rule set that minimizes the training error. In the exploitation mode, Fuzzy-UCS uses the evolved knowledge to infer the class of unlabeled examples. A concise description of the system is provided below.

A. Knowledge Representation

Fuzzy-UCS evolves a *population* [P] of classifiers which jointly represent the solution to a problem. Each classifier consists of a rule whose condition is in *conjunctive normal form* and a set of parameters. The fuzzy rule follows the structure

$$\textbf{IF } x_1 \text{ is } \widetilde{A_1^k} \text{ and } \cdots \text{ and } x_n \text{ is } \widetilde{A_n^k} \textbf{ THEN } c^k \textbf{ WITH } w^k \tag{1}$$

where each input variable x_i is represented by a disjunction of *linguistic terms* or *labels* $\widetilde{A_i^k} = \{A_{i1} \vee \cdots \vee A_{\mathrm{in}_i}\}$. In our experiments, all input variables share the same semantics, which are defined by means of triangular-shaped fuzzy membership functions. Note that this representation intrinsically permits generalization since each variable can take an arbitrary number of linguistic terms. The consequent of the rule indicates the class c^k which the rule itself predicts. w^k is a weight $(0 \leq w^k \leq 1)$ that denotes the soundness with which the rule predicts the class c^k. These types of rules with a weight in the consequent are known as fuzzy rules of type II [19].

The *matching degree* $\mu_{A^k}(e)$ of an example e with a classifier k is computed as follows. For each variable x_i, we compute the membership degree for each of its linguistic terms, and aggregate them by means of a T-conorm (disjunction). We enable the system to deal with missing values by considering that $\mu_{A^k}(e_i) = 1$ if the value e_i for the input variable x_i is not known. Then, the matching degree of the rule is determined by the T-norm (conjunction) of the matching degree of all the input variables. In our implementation, we used a *bounded sum* $(\min\{1, a+b\})$ as T-conorm and the *product* $(a \cdot b)$ as T-norm. Note that, if the fuzzy partition guarantees that the addition of all membership degrees is greater than or equal to 1—the membership functions used in our experiments satisfy this condition—, the selected T-norm and T-conorm allow for a maximum generalization. Therefore, an input variable x_i consisting of two consecutive linguistic terms will result in a matching degree of $\mu_{x_i}(e) = 1$ if the matching of e_i with both linguistic terms is greater than zero; thus, this choice supports the absence of the variable x_i.

Each classifier has four main parameters: 1) the fitness F, which estimates the accuracy of the rule; 2) the correct set size *cs*, which averages the sizes of the correct sets in which the classifier has participated (see Section III-B); 3) the experience *exp*, which computes the contributions of the rule to classify the input instances; and 4) the numerosity *num*, which counts the number of copies of the rule in the population.

B. Learning Interaction

The learning interaction is inherited from UCS (see Section II-B) and adapted to deal with fuzzy rules. For this purpose, three main differences with respect to UCS need to be considered: the *matching calculation*, the *rule structure*, and the *inference methodology*.

1) *Matching calculation.* In UCS, the attributes are represented by intervals $[l_i, u_i]$, and thus, a rule matches an input example if $\forall\ e_i : l_i \leq e_i \leq u_i$. Therefore, the matching function returns a binary output indicating whether the classifier matches the example e or not. In Fuzzy-UCS, a rule k matches the input example with a matching degree $\mu_{A^k}(e)$, where $0 \leq \mu_{A^k}(e) \leq 1$. High values of $\mu_{A^k}(e)$ indicate that the prediction of rule k is fairly accurate.
2) *Rule structure.* In UCS, a rule predicts a single class with a certain fitness or quality. Consequently, the population may contain two rules with the same antecedent advocating different classes. To avoid this situation, rules in Fuzzy-UCS maintain a weight for each class that indicates the soundness in which this class is predicted. The class advocated by the rule is the class with the maximum weight.
3) *Inference methodology.* In UCS, all the classifiers in [M] emit a fitness-weighted vote for the class they advocate, and the most voted class is chosen as the predicted output. In Fuzzy-UCS, different fuzzy-logic inference methods can be used to infer the class from the final fuzzy rule set [41]. Section III-E presents the three types of inference used.

The learning interaction of Fuzzy-UCS was redesigned considering these differences. First, the match set [M] is created with all the classifiers in [P] that have a matching degree $\mu_{A^k}(e)$ greater than zero.[1] Next, in exploration mode, the classifiers in [M] that advocate the class c form the correct set [C]. In exploit mode, the class is inferred using one of the three methodologies detailed in Section III-E, and no further action is taken.

If none of the classifiers in [C] match e with the *maximum matching degree*, the covering operator is triggered, which creates the classifier that maximally matches the input example. That is to say, for each attribute of the condition, we aggregate the linguistic term A_{ij} that maximizes the matching with the input value e_i. If e_i is not known, we randomly select a linguistic term and aggregate it to the attribute. Moreover, we introduce generalization by permitting the addition of other linguistic terms with probability $P_{\#}$. The initial values of the new classifiers are initialized according to the information provided by the current examples. Specifically, the fitness, the numerosity, and the experience are set to 1. The fitness of a new rule is set to 1 to give it opportunities to take over. Nonetheless, two important aspects should be noted. First, as the new classifiers participate in new match sets, their fitness and other parameters are quickly updated to their average values, and so, the initial value is not crucial. Second, as specified in Sections III-CF, the system prevents young classifiers from having a strong presence in the genetic selection, and protects them from an early deletion. At the end of the covering process, the new classifier is inserted in the population, deleting another one if there is not room for it.

[1]We do not require that rules have a matching degree greater than a certain threshold to be in [M], as sometimes done in regression [26]. In regression, the output is formed by means of aggregating rules with different actions. Thus, a minimum matching degree with the input may be required to participate in this process. However, in Fuzzy-UCS, the rules in [C] advocate the same class. In this way, Fuzzy-UCS avoids aggregating rules of different classes in the learning process, and so, a matching threshold appears to be less necessary.

C. Parameters Update

At the end of each learning iteration, Fuzzy-UCS updates the parameters of the rules in [M]. First, the experience of the rule is incremented according to the current matching degree:

$$\exp^k_{t+1} = \exp^k_t + \mu_{A^k}(e). \tag{2}$$

Next, the fitness is updated. For this purpose, each classifier internally maintains a vector of classes $\{c_1, \ldots, c_m\}$, each of them with an associated weight $\{v^k_1, \ldots, v^k_m\}$. Each weight v^k_j indicates the soundness with which rule k predicts class j for an example that fully matches this rule. These weights are incrementally updated during learning as explained as follows. The class c^k advocated by the rule is the class with the maximum weight v^k_j. Thus, given that the weights may change due to successive updates, the class that a rule predicts may also vary.

To update the weights, we first compute the sum of correct matchings cm^k for each class j:

$$cm^k_{j_{t+1}} = cm^k_{j_t} + m(k, j) \tag{3}$$

where

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

Then, cm^k_{j+1} is used to compute the weights v^k_{j+1}:

$$\forall\ j : v^k_{j_{t+1}} = \frac{cm^k_{j_{t+1}}}{\exp^k_{t+1}}. \tag{5}$$

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

The fitness is then computed from the weights with the aim of favoring classifiers that match examples of a single class. To carry this out, we use the following formula [42]:

$$F^k_{t+1} = v^k_{\max_{t+1}} - \sum_{j|j\neq\max} v^k_{j_{t+1}} \tag{6}$$

where we subtract the values of the other weights from the weight with maximum value $v^k_{\max}$. The fitness F^k is the value used as the weight w^k of the rule [see (1)]. Note that this formula can result in classifiers with zero or negative fitness (for example, if the number of classes is greater than 2 and the class

weights are equal). Finally, the correct set size of all the classifiers in [C] is calculated as the arithmetic average of the sizes of all the correct sets in which the classifier has participated.

Finally, the rule k predicts the class c with the highest weight associated v_c^k. Thus, the class predicted is not fixed when the rule is created, and can change as the parameters of the rule are updated (especially during the first parameters updates).

D. Discovery Component

Fuzzy-UCS uses a steady-state niched *genetic algorithm* (GA) [43] to discover new promising rules. The GA is applied to the classifiers that belong to [C]. Thus, the niching is intrinsically provided since the GA is applied to rules that match the same input with a degree greater than zero and advocate the same class.

The GA is triggered when the average time from its last application upon the classifiers in [C] exceeds the threshold θ_{GA}. It selects two parents p_1 and p_2 from [C] using proportionate selection [28], where the probability of selecting a classifier k is

$$p_{\mathrm{sel}}^k = \frac{(F^k)^\nu \cdot \mu_{A^k}(e)}{\sum_{i\in[C]|F^i\geq 0} (F^i)^\nu \cdot \mu_{A^k}(e)} \tag{7}$$

where $\nu > 0$ is a constant that fixes the pressure toward maximally accurate rules (in our experiments, we set $\nu = 10$). Rules with negative fitness are not considered for selection. The two parents are copied into offspring ch_1 and ch_2, which undergo crossover and mutation with probabilities χ and μ respectively. The crossover operator crosses the antecedents of the rules by two points. The mutation operator checks whether each variable has to be mutated with probability μ. If so, 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 be applied only to variables that do not have all the linguistic terms. Contraction selects a linguistic term represented in the variable and removes it; so, it can be applied only to variables that have more than one linguistic term. By doing so, we avoid generating rules that do not match any example. Shift changes a linguistic term for its immediate inferior or superior.

The new offspring are introduced into the population. First, each classifier is checked for subsumption [4] with their parents. Subsumption is a mechanism that prevents the creation of classifiers with specific conditions if there are more general and accurate classifiers in the population that cover the same region of the feature space. The process works as follows. If any parent's condition subsumes the condition of the offspring (i.e., the parent has, at least, the same linguistic terms per variable than the child), and this parent is highly accurate $\left(F^k > F_0^k\right)$ and sufficiently experienced $\left(\exp^k > \theta_{\mathrm{sub}}\right)$, the offspring is not inserted and the numerosity of the parent is increased by one. Otherwise, we check [C] for the most general rule that can subsume the offspring. If no subsumer can be found, the classifier is inserted in the population.

If the population is full, excess classifiers are deleted from [P] with probability proportional to the correct set size estimate cs. Moreover, if the classifier is sufficiently experienced $\left(\exp^k > \theta_{\mathrm{del}}\right)$ and the power of its fitness $(F^k)^\nu$ is significantly lower than the average fitness of the classifiers in [P] ($(F^k)^\nu < \delta F_{[P]}$ where $F_{[P]} = (1/N)\sum_{i\in[P]}(F^i)^\nu$), its deletion probability is further increased. That is, each classifier has a deletion probability p_k of

$$p_k = \frac{d_k}{\sum_{\forall\, j\in[P]} d_j} \tag{8}$$

where

$$d_k = \begin{cases} \frac{cs\cdot \mathrm{num}\cdot F_{[P]}}{(F^k)^\nu}, & \text{if } \exp^k > \theta_{\mathrm{del}} \text{ and } (F^k)^\nu < \delta F_{[P]} \\ cs\cdot \mathrm{num}, & \text{otherwise.} \end{cases} \tag{9}$$

Thus, the deletion algorithm balances the classifier's allocation in the different correct sets by pushing toward deletion of rules belonging to large correct sets. At the same time, it favors the search toward highly fit classifiers, since the deletion probability of rules whose fitness is much smaller than the average fitness is increased.

E. Fuzzy-UCS in Test Mode

The aim of Fuzzy-UCS is to evolve a minimum set of maximally accurate rules that cooperate to cover all the input space. To achieve high classification accuracy, we need to define effective reasoning methods that use the information of the rule set to infer the class of new input examples. As these reasoning methodologies may not use all the rules in the inference process, rule set reduction techniques can be applied to remove the rules that are not considered for the reasoning technique. Herein, we discuss two different inference schemes. Furthermore, we present a reduction method for each one of these inference techniques that permits to reduce the number of rules in the final population without decreasing training accuracy. Finally, we also introduce a third rule set reduction mechanism which allows for higher reductions but does not guarantee that the reduced rule set results in the same training performance as the original.

1) Class Inference: Once Fuzzy-UCS has evolved a population of highly general and accurate rules, this population is used to infer the class of new examples. Given a new unlabeled instance e, several rules predicting different classes can match (with different degrees) this instance. Thus, the knowledge contained in the set of matching classifiers has to be combined to decide the most likely output. For this purpose, several reasoning methodologies have been analyzed in for fuzzy-rule based systems [41], [44]. Here, we adapt two inference approaches to Fuzzy-UCS. In both cases, only experimented rules $\left(\exp^k > \theta_{\mathrm{exploit}}\right)$ are considered in the inference, where $\theta_{\mathrm{exploit}}$ is a user-set parameter that indicates the minimum experience that a rule must have to participate in the inference process.

Weighted average inference. In this approach, all the experienced rules vote to infer the output. Each rule k emits a vote

v_k for class j it advocates, where $v_k = F^k \cdot \mu_{A^k}(e)$. The votes for each class j are added:

$$\forall\, j : \text{vote}_j = \sum_{k|c^k=j}^{N} v_k \tag{10}$$

and the most-voted class is returned as the output.

Action winner inference. This approach selects the rule k that maximizes $\mu_{A^k}(e) \cdot F^k$, and chooses the class of the rule as output [25]. Thus, the knowledge of overlapping rules is not considered in this inference scheme.

2) Rule Set Reduction: At the end of the learning process, the population is reduced to obtain a minimum set of rules. We designed three types of reduction, which use one of the inference schemes presented above.

Reduction based on weighted average. Under the weighted average scheme, we reduce the final population by removing all the rules that a) are not experienced enough $(\exp > \theta_{\text{exploit}})$ or b) have zero or negative fitness.

Reduction based on action winner. If action winner inference is used, only rules that maximize the prediction vote for a training example are necessary. Thus, after training, this reduction scheme infers the output for each training example. The rule that maximizes the vote v_j for each example is copied to the final population.

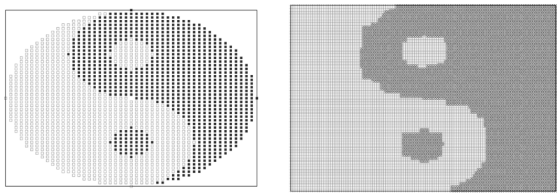
Reduction based on the most numerous and fittest rules. This reduction tries to minimize the rule set size by selecting the most numerous and accurate rules for the final population. The methodology is a hybrid of the previous approaches. The reduction process is analogous to the reduction based on action winner, but now, the rule k that maximizes $F^k \cdot \mu_{A^k}(e) \cdot \text{num}^k$ for each input example is copied to the final population. By including the numerosity in the vote, we favor the most numerous and accurate rules. As this reduction may copy overlapping rules into the final population, weighted average is used to infer the class of a new example.

In Section III-F, we will analyze the differences between these inference and reduction techniques. To facilitate the notation, these schemes will be addressed as: weighted average inference (wavg), action winner inference (awin), and most numerous and fittest rules inference (nfit).

F. Interpretability of Weighted Fuzzy Classification Rules

The inclusion of weights to express the importance degree of fuzzy classification rules, as Fuzzy-UCS does, has been studied from different points of view. In general, the inclusion of additional parameters in the fuzzy rules, such as weights, results in a decrease of the interpretability degree of the learned knowledge. However, it is interesting to discuss to which degree interpretability is lost, and to what extend accuracy can be improved.

Nauck and Kruse [45] analyze the effect of rule weights in fuzzy rule-based systems for regression problems. They argue that rule weights may hinder the interpretability of such systems, showing that rule weights could be replaced by the modification of the membership functions of fuzzy rules. In fact, the use of rules with different importance degrees could hinder the interpolative reasoning made by the inference engine when continuous outputs are returned in regression problems.



(a) Domain (b) Interval-based UCS

Fig. 2. (a) Domain of the tao problem and (b) decision boundaries obtained by UCS. UCS achieved 99.80% training accuracy and evolved 1230 rules.

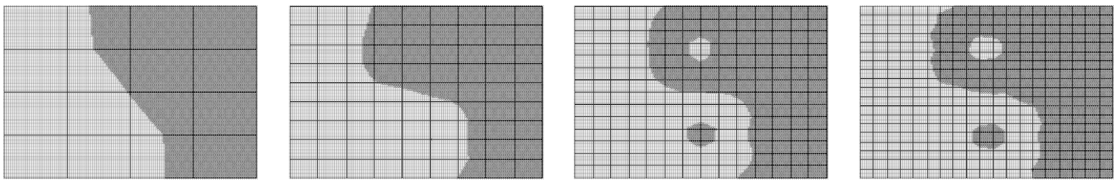
In data classification tasks, however, other analyses emphasize the benefit of using rule weights as a mechanism to improve the accuracy while preserving good interpretability. Indeed, Ishibuchi and Nakashima [46] discuss the importance of rule weights (certainty degrees) from a completely different point of view. They point out that the use of weights allows the system to reach a high degree of accuracy with fixed membership functions since these certainty degrees effectively modify the decision areas. Therefore, they advocate the use of weights to improve accuracy instead of modifying the membership functions of the given linguistic terms. They also show that weights play an important role when the fuzzy rule-based classification system collects rules of different generality degrees, as in the case of our Fuzzy-UCS algorithm.

Finally, it is worth mentioning that the use of weighted fuzzy rules in classification is a common practice frequently referred to in specialized literature [41], [42], [46]–[48].

IV. Knowledge Representation and Decision Boundaries

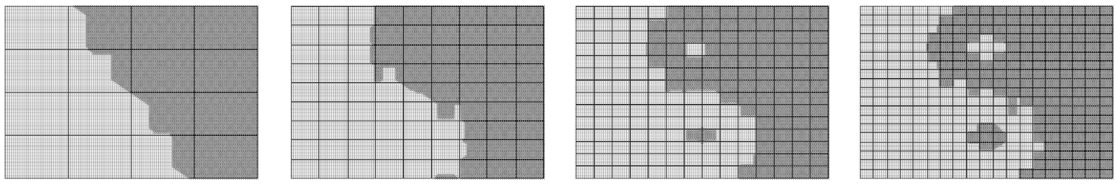
So far, we have described the Fuzzy-UCS classifier system with a *descriptive* or *linguistic* representation of fuzzy rules. Linguistic rules are highly interpretable since they share common semantics; however, as this representation implies the discretization of the feature space, a single rule may not have the required granularity to define the class boundary of a given domain accurately. Thus, Fuzzy-UCS evolves a set of overlapping fuzzy-rules around the decision boundaries which match examples of different classes, and the output depends on how the reasoning mechanism combines these overlapping rules. Fuzzy-UCS includes three inference and reduction schemes which lead to a trade-off between the amount of information used for the inference process (i.e., the precision of the prediction) and the size of the rule set. Consequently, not only the linguistic representation but also the inference and reduction schemes chosen may impose a maximum limit on the accuracy rate that the system can reach.

This section studies the interpretability-performance trade-off in Fuzzy-UCS. We illustrate how the three inference schemes approximate the decision boundaries of an artificial problem and compare their differences in terms of accuracy and interpretability. Moreover, we empirically analyze the sensitivity of Fuzzy-UCS to different configurations, highlighting its robustness to most of the configuration parameters.



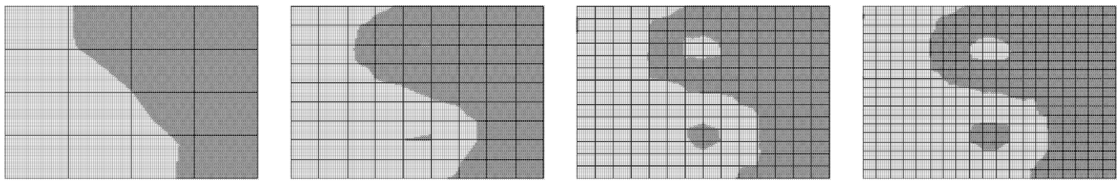
(a) 5 labels (b) 10 labels (c) 15 labels (d) 20 labels

Fig. 3. Decision boundaries obtained with weighted average inference and (a) 5, (b) 10, (c) 15, and (d) 20 linguistic terms per variable. Fuzzy-UCS achieved {82.95%, 91.85%, 96.68%, 97.15%} training accuracy and evolved {112, 441, 618, 763} rules, respectively.



(a) 5 labels (b) 10 labels (c) 15 labels (d) 20 labels

Fig. 4. Decision boundaries obtained with action winner inference and (a) 5, (b) 10, (c) 15, and (d) 20 linguistic terms per variable. Fuzzy-UCS achieved {83.24%, 91.19%, 94.74%, 95.57%} training accuracy and evolved {17, 78, 144, 200} rules, respectively.



(a) 5 labels (b) 10 labels (c) 15 labels (d) 20 labels

Fig. 5. Decision boundaries obtained with most numerous and fittest rules inference and (a) 5, (b) 10, (c) 15, and (d) 20 linguistic terms per variable. Fuzzy-UCS achieved {88.31%, 91.85%, 96.68%, 97.15%} training accuracy and evolved {15, 30, 52, 65} rules, respectively.

A. Decision Boundaries: Study on an Artificial Domain

We first analyzed the three inference schemes of Fuzzy-UCS on a case study. We also included UCS with interval-based representation [5], [33] in the analysis. We graphically studied how the three inference schemes approximated the decision boundaries of an artificially designed domain with respect to interval-based UCS. We chose a two-dimensional problem to facilitate the visualization: the tao problem [49] [see Fig. 2(a)]. The input range of the two variables of the tao problem is $[-6, 6]$. This problem presents curved-shaped boundaries, whose approximation poses a challenge to the linguistic fuzzy representation. Moreover, we compared the training accuracies, as well as the size of the evolved rule set. This analysis was restricted to the features of the tested problem, and only estimated the training error; thus, our aim was not to extract general conclusions, but to provide an intuitive visualization of the decision boundaries defined by each inference scheme. This analysis is complemented in Section IV-B, where the three inference schemes are compared in a set of real-world problems.

We configured UCS as: $\mathrm{numIter} = 100{,}000$, $N = 6400$, $\mathrm{acc}_0 = 0.99$, $\nu = 10$, $\{\theta_{\mathrm{GA}}, \theta_{\mathrm{del}}, \theta_{\mathrm{sub}}\} = 50$, $\chi = 0.8$, $\mu = 0.04$, $\delta = 0.1$, $r_0 = 0.2$. Similar parameters were used for Fuzzy-UCS: $N = 6400$, $F_0 = 0.99$, $\nu = 10$, $\{\theta_{\mathrm{GA}}, \theta_{\mathrm{del}}, \theta_{\mathrm{sub}}\} = 50$, $\theta_{\mathrm{exploit}} = 10$, $\chi = 0.8$, $\mu = 0.6$, $\delta = 0.1$, and $P_{\#} = 0.2$, We initialized the population with quite specific rules since the problem has only two dimensions and a high density of instances. Fig. 2(b) depicts the boundaries evolved by interval-based UCS. Figs. 3–5 report the decision boundaries for Fuzzy-UCS with weighted average inference (wavg), action winner inference (awin), and most numerous and fittest rules inference respectively (nfit). In each case, we experimented with 5, 10, 15, and 20 linguistic terms per variable; the grid in the plots indicates the partitions in the feature space made by the cross-points of the triangular membership functions associated to the different fuzzy sets. The training accuracies achieved and the sizes of the populations evolved are summarized in the captions of the figures. The results are averages over ten runs with different seeds.

Several observations can be drawn from the evolved decision boundaries. First, the results show the generalization capabilities of all learners. The rules tend to expand as much as possible while they are accurate, covering regions in the feature space where there are no examples. This generalization pressure is mostly due to subsumption, which replaces the offspring for more general and accurate rules when possible. Thus, this operator gives more strength to highly general and accurate rules.

Interval-based UCS reached the maximum accuracy among all learners. It evolved a population consisting of 1230 rules which accurately defined the decision boundaries [see Fig. 2(b)], with 99.8% training accuracy. The accuracy obtained by Fuzzy-UCS depended on the number of linguistic terms per variable (see the models built in Figs. 3–5). With 5 linguistic labels per variable, Fuzzy-UCS could not discover

TABLE I
PROPERTIES OF THE DATASETS. THE COLUMNS DESCRIBE: THE IDENTIFIER OF THE DATASET (ID.) THE NAME OF THE DATASET (DATASET), THE NUMBER OF INSTANCES (#INST), THE TOTAL NUMBER OF FEATURES (#FEA), THE NUMBER OF REAL FEATURES (#RE), THE NUMBER OF INTEGER FEATURES (#IN), THE NUMBER OF NOMINAL FEATURES (#NO), THE NUMBER OF CLASSES (#CL), THE PROPORTION OF INSTANCES OF THE MINORITY CLASS (%MIN), THE PROPORTION OF INSTANCES OF THE MAJORITY CLASS (%MAJ), THE PROPORTION OF INSTANCES WITH MISSING VALUES (%MISINST), AND THE PROPORTION OF FEATURES WITH MISSING VALUES (%MISATT)

Id.	dataset	#Inst	#Fea	#Re	#In	#No	#Cl	%Min	%Maj	%MisInst	%MisAtt
ann	Annealing	898	38	6	0	32	5	0.9	76.2	0	0
aut	Automobile	205	25	15	0	10	6	1.5	32.7	22.4	28
bal	Balance	625	4	4	0	0	3	7.8	46.1	0	0
bpa	Bupa	345	6	6	0	0	2	42	58	0	0
cmc	Contraceptive method choice	1473	9	2	0	7	3	22.6	42.7	0	0
col	Horse colic	368	22	7	0	15	2	37	63	98.1	95.5
gls	Glass	214	9	9	0	0	6	4.2	35.5	0	0
h-c	Heart-c	303	13	6	0	7	2	45.5	54.5	2.3	15.4
h-s	Heart-s	270	13	13	0	0	2	44.4	56.6	0	0
irs	Iris	150	4	4	0	0	3	33.3	33.3	0	0
pim	Pima	768	8	8	0	0	2	34.9	65.1	0	0
son	Sonar	208	60	60	0	0	2	46.67	53.33	0	0
tao	Tao	1888	2	2	0	0	2	50	50	0	0
thy	Thyroid	215	5	5	0	0	3	14	60	0	0
veh	Vehicle	846	18	18	0	0	4	23.5	25.8	0	0
wbcd	Wisc. breast-cancer	699	9	0	9	0	2	34.5	65.5	2.3	11.1
wdbc	Wisc. diagnose breast-cancer	569	30	30	0	0	2	37.3	62.7	0	0
wne	Wine	178	13	13	0	0	3	27	39.9	0	0
wpbc	Wisc. prognostic breast-cancer	198	33	33	0	0	2	23.7	76.3	2	3
zoo	Zoo	101	17	0	1	16	7	4	40.6	0	0

the two inner concepts of the tao problem regardless of the inference method used. The models only defined one linear class boundary that did not fit the curved boundary of the domain accurately. As the number of linguistic terms per variable increased, the boundaries were defined more accurately. With 20 linguistic terms per variable, the three types of inference achieved high training performances.

The models evolved by Fuzzy-UCS with the three types of inference differed in the shape of the decision boundaries and the rule set size. Weighted average inference defined smooth boundaries which resulted from the vote of several overlapping rules (see Fig. 3). However, it maintained a high number of rules in the final population. Action winner inference created more reduced rule sets, but the boundaries defined were more abrupt. Note that the decision boundaries followed the partitions produced by the fuzzy membership functions, especially when 15 and 20 linguistic terms were used. This is because only the rules that maximized the product of $\mu_{A^k}(e) \cdot F$ were kept in the final population. Most numerous and fittest rules inference evolved the most compact rule sets. Furthermore, the boundaries were smoother than those obtained with action winner scheme. This type of inference maintained the most numerous and accurate rules in the final population. As this process could insert overlapping rules into the final population, the weighted average inference was used to infer the class, thus forwarding the interpolative reasoning. For this reason the decision boundaries were not as abrupt as those evolved by the action winner inference.

B. Comparison Among the Three Inference Schemes

This section furthers the study on the three types of inference of Fuzzy-UCS. Specifically, we examine the trade-off between precision and rule set size already pointed out in the previous section for the three types of inference methodologies of Fuzzy-UCS.

1) Methodology: We selected a collection of 20 real-world data sets whose characteristics are summarized in Table I. All the data sets were obtained from the UCI Repository [50], except for *tao*, which was selected from a local repository [49].

The performance of the methods was measured by the test accuracy rate, i.e., the proportion of correct predictions on previously unseen instances. We collected the evolved rule set sizes to compare the interpretability of the three configurations of Fuzzy-UCS. To obtain reliable estimates of these metrics, we used a ten-fold cross validation procedure [51].

The results were statistically analyzed following the recommendations pointed out in [52]. In all the analysis we used nonparametric statistical tests to compare the results obtained by the different learning algorithms. Parametric tests require that the input data (in our case, the tables of results) satisfy strong conditions, and the tests to check these conditions need large amounts of data (i.e., large number of data sets) to be effective [53]. For this reason, nonparametric tests are recommended [52], since they relax the requirements on the input data.

We applied multiple-comparison statistical procedures to test the null hypothesis that all the learning algorithms performed

equivalently on average. Specifically, we used Friedman's test [54], [55], a nonparametric equivalent of the repeated-measures ANOVA [56]. If Friedman's test rejected the null hypothesis, we used the nonparametric Nemenyi test [57] to compare all learners to each other. The Nemenyi test defines that two results are significantly different if the corresponding average rank differs by at least a critical difference CD computed as

$$\mathrm{CD} = q_\alpha \sqrt{\frac{n_\ell(n_\ell+1)}{6n_{\mathrm{ds}}}} \tag{11}$$

where n_ℓ and n_{ds} are the number of learners and the number of data sets respectively, and q_α is the critical value based on the Studentized range statistic [53]. The Nemenyi test is said to be quite conservative, especially when a large number of learners are compared, so that it might not detect some existent differences between learners. Therefore, we complemented the statistical analysis by comparing the performance of each pair of learners by means of the nonparametric Wilcoxon signed-ranks test [58]. The approximate p-values resulting from the pairwise analysis, calculated as indicated in [53], were provided in the analysis.

The configuration used for Fuzzy-UCS was the same as in Section IV-A, but we set $P_\# = 0.6$ to initialize the population with more general rules. In the remainder of this paper, we refer to this configuration as the default configuration, since it uses equivalent parameter values to those usually set for XCS and UCS. Moreover, we fixed the number of linguistic terms to 5. We did not consider a larger number of linguistic terms since it could hinder the interpretability desired in a linguistic representation.

2) Results: Table II shows the test accuracy and the number of rules of the models evolved by Fuzzy-UCS with each inference scheme. The two last rows supply the average rank and the position of each algorithm in the ranking. The ranks were calculated as follows. For each data set, we ranked the learning algorithms according to their performance; the learner with highest accuracy held the first position, whilst the learner with the lowest accuracy held the last position of the ranking. If a group of learners had the same performance, we assigned the average rank of the group to each of the learners in the group. The same process was followed with the number of rules, but in this case, the model with the lowest number of rules held the first position of the ranking.

The multiple-comparison test rejected the hypothesis that all learners performed the same on average at a significance level of 0.0001. The post-hoc Nemenyi test, at a significance level of 0.10, identified two inference schemes that performed equivalently: action winner and most numerous and fittest rules inferences. The weighted average inference resulted in the most accurate models. The same significant differences were found with the pairwise statistical analysis.

Friedman's test also rejected the hypothesis that the population sizes were equivalent on average at a significance level of 0.0001 (see Table II). The post-hoc Nemenyi test, at a significance level of 0.10, supported the hypothesis that the four learners evolved populations with significantly different sizes.

TABLE II
COMPARISON OF THE TEST ACCURACY AND THE NUMBER OF RULES OF THE MODELS CREATED BY FUZZY-UCS WITH WEIGHTED AVERAGE INFERENCE (WAVG), FUZZY-UCS WITH ACTION WINNER INFERENCE (AWIN), AND FUZZY-UCS WITH MOST NUMEROUS AND FITTEST RULES INFERENCE (NFIT)

	Performance			**Number of Rules**		
	wavg	**awin**	**nfit**	**wavg**	**awin**	**nfit**
ann	98.85	97.39	98.61	2769	75	36
aut	74.42	67.42	69.32	3872	114	74
bal	88.65	84.40	83.40	1212	114	75
bpa	59.82	59.42	58.93	1440	73	39
cmc	51.72	49.67	49.42	1881	430	271
col	85.01	82.46	78.50	4135	154	81
gls	60.65	57.21	57.43	2799	62	36
h-c	84.39	82.62	82.05	3574	113	46
h-s	81.33	80.78	78.11	3415	117	62
irs	95.67	95.47	93.73	480	18	7
pim	74.88	74.11	74.32	2841	192	62
son	80.78	73.71	71.66	3042	178	160
tao	81.71	83.02	87.53	111	19	14
thy	88.18	89.49	91.25	1283	37	11
veh	67.68	65.35	65.34	3732	332	147
wbcd	96.01	95.73	95.29	3130	138	28
wdbc	95.20	94.61	94.51	5412	276	101
wne	94.12	94.86	91.82	3686	95	26
wpbc	76.06	76.05	71.69	3772	156	115
zoo	96.50	94.78	95.90	773	16	10
Rank	*1.25*	*2.2*	*2.55*	*3*	*2*	*1*
Pos	*1*	*2*	*3*	*3*	*2*	*1*

The pairwise comparisons yielded the same conclusions. In fact, a simple quantitative analysis highlighted the differences in the population sizes. Fuzzy-UCS with weighted average inference built populations that consisted of thousands of rules. Consequently, although using a linguistic representation, this high number of rules hampered the interpretability of the rule set. The other two types of inference of Linguistic Fuzzy-UCS, especially the most numerous and fittest rules inference, resulted in populations with a moderate number of rules. Fuzzy-UCS with most numerous and fittest rules inference built populations that ranged from tens of to few hundreds of rules.

These results showed the performance-interpretability trade-off in Fuzzy-UCS already pointed out in the previous section. Weighted average inference significantly outperformed the other two inference schemes since it combined the knowledge of all experienced rules in the final population. As shown in the case study of the previous section, this allowed Fuzzy-UCS to fit complex boundaries even though the fuzzy representation made a discretization of the feature space. Fuzzy-UCS could approximate these boundaries by means of evolving a set of partially overlapping fuzzy rules. However, the interpretability of the rule set was degraded by the high number of rules. The other two inference schemes considerably improved the readability, since they produced large reductions of the rule set. Nonetheless, this went against the test performance, which

TABLE III
CONFIGURATIONS USED TO TEST THE SENSITIVITY OF FUZZY-UCS TO CONFIGURATION PARAMETERS

	Cp	N=6400, F_0 = 0.99, ν = 10, $\{\theta_{GA}, \theta_{del}, \theta_{sub}\}$ = 50, $\theta_{exploit}$ = 10, χ = 0.8, μ = 0.1, δ=0.1, and $P_{\#} = 0.2$
$P_{\#}$	**C1**	$P_{\#} = 0.2$
	C2	$P_{\#} = 0.4$
ν	**C3**	$\nu = 1$
	C4	$\nu = 5$
$\theta_{GA}, \theta_{del}, \theta_{sub}$	***C5***	$\theta_{GA} = \theta_{del} = \theta_{sub} = 100$ and $numIter = 100,000$
	C6	$\theta_{GA} = \theta_{del} = \theta_{sub} = 200$ and $numIter = 100,000$
	C7	$\theta_{GA} = \theta_{del} = \theta_{sub} = 100$ and $numIter = 200,000$
	C8	$\theta_{GA} = \theta_{del} = \theta_{sub} = 200$ and $numIter = 400,000$
δ	***C9***	$\delta = 1$

was significantly surpassed by the weighted average inference scheme.

C. Sensitivity of Fuzzy-UCS to Configuration Parameters

In common with many competitive Michigan-style LCSs, Fuzzy-UCS has several configuration parameters, which enable it to adjust the behavior of the system to evolve models of maximal quality for particular problems. At first glance, choosing a correct configuration may seem a crucial task only suitable for expert users. Nonetheless, several analyses identified the robustness of Michigan-style LCSs to the majority of configuration parameters. Actually, most of the applications of Michigan-style LCSs used the same default parameters to solve pattern recognition problems [5], [9], [10], [49], [59]. We consider that this robustness is also present in Fuzzy-UCS. Therefore, we used the same default configuration to solve the collection of real-world problems. In this section, we empirically show the behavior of Fuzzy-UCS with different configurations and relate this analysis to theoretical and empirical studies of the sensitivity of LCSs—particularly XCS and UCS—to configuration parameters. Theoretical and empirical analyses of the sensitivity of LCSs[2] to configuration parameters detected two crucial parameters: (i) population initialization [60], and (ii) fitness pressure [61]. Moreover, facetwise models were derived to explain how the genetic algorithm could maintain the different niches of the system [40]. The other parameters became nearly constant. Herein, we empirically show this behavior for Fuzzy-UCS. For this purpose, we analyzed the accuracy and size of the models evolved by Fuzzy-UCS related to the changes of four parameters or groups of parameters: (1) rules generalization in initialization, i.e., $P\#$; (2) fitness pressure, i.e., ν; (3) setting of the genetic algorithm, i.e., θ_{GA}, θ_{del}, and θ_{sub}; and (4) deletion pressure, i.e., δ. We compared different configuration settings to the default configuration (Cp). Given the large number of configurations tested, we used a reduced collection of data sets: *bal*, *bpa*, *gls*, *h-s*, *irs*, *pim*, *tao*, *thy*, *veh*, *wbcd*, *wdbc*, and *wne*.

[2]These analyses refer to XCS and UCS, but could be easily extended to other Michigan-style LCSs.

Table III summarizes the different configurations and the changes that they introduced with respect to the default configuration. Table IV provides the average rank of the model's accuracy and size for each configuration and inference scheme. We divided the configuration settings into four groups, and each group was compared to the default configuration. The best ranked configurations for each comparison are marked in bold. The symbol $\ominus$ indicates that the corresponding configuration significantly degraded the results obtained with the best configuration according to a Bonferroni-Dunn test at $\alpha = 0.1$ [62].

The results show that the generalization in the initial population is essential to the success of Fuzzy-UCS, supporting the theoretical analyses in the literature [60]. For all the inference schemes, configurations Cp and $C2$ (i.e., $P_{\#} = \{0.6, 0.4\}$) were statistically equivalent, on average, and significantly better than $C1$ (i.e., $P_{\#} = 0.2$) in terms of accuracy. In terms of model size, the following significant differences were found: (i) for weighted average inference, Cp and $C1$ evolved the smallest rule sets; (ii) for action winner and most numerous and fittest rules inference, $C1$ created significantly larger rule sets than Cp and $C2$. The last point can be easily explained as follows. As $C1$ used a low value of $P_{\#}$, final populations contained more specific classifiers than populations created with Cp and $C2$. Action winner and most numerous and fittest rules schemes kept in the final populations only the classifiers that maximized the product of fitness and matching degree with a training instance. As classifiers were more specific, a larger number of them were placed in the final population. With weighted average, the biggest population sizes were obtained with $C2$. This could be due to the existence of slightly general classifiers that were all maintained in the final population.

The second comparison shows the negative influence of having low fitness pressure. In terms of accuracy, better results were obtained as the fitness pressure increased (i.e., ν took higher values). Population sizes varied with the fitness pressure depending on the inference scheme. For weighted average inference, Cp led to the significantly smaller rule sets. This is because the fitness pressure drove toward a highly general and accurate set of rules. For the other two inference schemes, configuration $C1$ resulted in the significantly smaller rule sets. That is, as the fitness pressure was low, populations were full

TABLE IV
COMPARISON OF THE SENSITIVITY OF FUZZY-UCS TO CONFIGURATION PARAMETERS. EACH CELL SHOWS THE AVERAGE RANK OF EACH CONFIGURATION FOR A GIVEN INFERENCE SCHEME. THE BEST RANKED METHOD IS IN BOLD. THE SYMBOL $\ominus$ INDICATES THAT THE CORRESPONDING METHOD SIGNIFICANTLY DEGRADES THE RESULTS OBTAINED WITH THE BEST RANKED METHOD

		Performance			Rule set size		
		wavg	**awin**	**nfit**	**wavg**	**awin**	**nfit**
$P_\#$	**Cp**	1.83	1.83	**1.83**	**1.67**	**1.50**	1.75
	C1	2.42 $\ominus$	2.50 $\ominus$	2.25 $\ominus$	**1.75**	2.92 $\ominus$	3.00 $\ominus$
	C2	**1.75**	**1.67**	1.92	2.58 $\ominus$	1.58	**1.25**
ν	**Cp**	**1.25**	**1.42**	**1.42**	**1.08**	2.33 $\ominus$	2.67 $\ominus$
	C3	2.83 $\ominus$	2.75 $\ominus$	2.83 $\ominus$	3.00 $\ominus$	**1.25**	**1.17**
	C4	1.92	1.83	1.75	1.92 $\ominus$	2.42 $\ominus$	2.17 $\ominus$
θ_{GA}, θ_{del} & θ_{sub}	**Cp**	**1.92**	**2.13**	**2.13**	3.42 $\ominus$	3.17	3.17
	C5	4.00 $\ominus$	3.42	3.58 $\ominus$	3.42 $\ominus$	3.50	2.92
	C6	4.33 $\ominus$	4.63 $\ominus$	4.17 $\ominus$	**1.75**	2.83	**2.58**
	C7	2.33	2.25	2.29	3.42 $\ominus$	3.33	3.17
	C8	2.42	2.58	2.83	3.00	**2.17**	3.17
δ	**Cp**	**1.25**	1.54	**1.50**	**1.33**	1.75	1.75
	C9	1.75	**1.46**	**1.50**	1.67	**1.25**	**1.25**

of over-general rules, which were kept in the final populations in detriment to fitter and more specific classifiers.

The third comparison shows the influence of the parameters related to the genetic algorithm, i.e., θ_{GA}, θ_{del}, and θ_{sub}. Initial intuition indicates that, if all niches receive the same number of genetic opportunities, the quality of the final models should remain the same. To test this, configurations $C7$ and $C8$ set $\theta_{\mathrm{GA}} = \theta_{\mathrm{del}} = \theta_{\mathrm{sub}} = \{100, 200\}$ and increased $\mathrm{numIter} = \{200000, 400000\}$ respectively. In this way, all niches received approximately the same number of genetic events. Configurations $C5$ and $C6$ fixed $\theta_{\mathrm{GA}} = \theta_{\mathrm{del}} = \theta_{\mathrm{sub}} = \{100, 200\}$ but maintained the same number of iterations as Cp. So, we expected that the quality of the models evolved by $C5$ and $C6$ was significantly lower than the quality of the models created by the three other configurations. This hypothesis was clearly supported by the experimental analysis, which showed that Cp, $C7$, and $C8$ resulted in the most accurate models. Moreover, significant differences on the population sizes were only found for the weighted average inference. The multiple comparison test detected that the smaller models were created with configurations $C6$ and $C8$, the two configurations in which the period of application of the GA was higher.

Finally, the fourth comparison highlights the robustness of Fuzzy-UCS to the deletion pressure toward unfit classifiers, that is, the parameter δ. The pairwise analysis indicated that the hypothesis that configurations Cp and $C9$ are equivalent could not be rejected, according to a Wilcoxon signed-ranks test at $\alpha = 0.05$.

The study conducted in this section empirically showed that there are two crucial parameters to guarantee the success of Fuzzy-UCS: generalization in initialization $P_\#$ and fitness pressure ν. Changing the setting of the other parameters had little effect on Fuzzy-UCS behavior. We acknowledge that better results could be individually obtained if we tuned Fuzzy-UCS for each particular problem. Nonetheless, as we are interested in robust systems that perform well on average, we use the default configuration for all the experiments in Section V.

V. COMPARISON OF FUZZY-UCS TO SEVERAL MACHINE LEARNING TECHNIQUES

In this section, we study whether the behavior of Fuzzy-UCS is comparable to some of the most-used machine learning techniques. For this purpose, we compared Fuzzy-UCS to two sets of learners: fuzzy rule-based learners and "nonfuzzy" (crisp) learners. With the former comparison, we analyzed the behavior of Fuzzy-UCS with respect to other techniques that use the same representation, which may limit the maximum performance that can be achieved in certain domains. With the latter comparison, we study whether, even with the limitations that may impose the fuzzy representation, Fuzzy-UCS is competitive with a large number of the most representative learners, regardless of the knowledge representation they use. Below, we first present the experimental methodology, and then compare Fuzzy-UCS to the other learners.

A. Experimental Methodology

The methodology followed is similar to the one presented in the previous section. We selected the same collection of 20 real-world problems, whose characteristics are summarized in Table I. The experiments ran on a ten-fold cross validation, and the test accuracy rate was used to measure the performance of the different learners. The performance of Fuzzy-UCS was compared with a large variety of learning algorithms, which we organized in two groups. The first group consisted of the following fuzzy rule-based classification systems: Fuzzy GP, Fuzzy GAP, Fuzzy SAP, Fuzzy AdaBoost, Fuzzy LogitBoost, and Fuzzy MaxLogitBoost. Fuzzy GP [63]–[65] is a genetic programming algorithm that builds a fuzzy classifier for each class of the domain by searching for a tree that relates the input and the output variables as accurately as possible. Fuzzy GAP [63],

[64] works similarly to Fuzzy GP, but the optimization system is a hybrid between genetic algorithms and genetic programming. Fuzzy SAP [65] combines genetic operators with simulated annealing [66] to create data models similar to those built by Fuzzy GP and Fuzzy GAP. Fuzzy AdaBoost [47] is a modification of the boosting algorithm AdaBoost [67] to deal with fuzzy rules. Fuzzy LogitBoost [48] and Fuzzy MaxLogitBoost [68] are boosting algorithms that iteratively invoke a genetic algorithm to extract simple fuzzy rules that are combined to decide the output of new examples. The basic difference between both algorithms is that Fuzzy MaxLogitBoost may reject a new rule provided by the genetic algorithm if it does not improve the expected global performance. All these methods were run using KEEL [69]. We followed the recommended parameter values given in the KEEL platform to configure the methods [69], which also corresponded to the settings used in the bibliography of these methods. We only changed the maximum population size of AdaBoost, LogitBoost, and MaxLogitBoost. We tried population sizes of $N = \{8,25,50,100\}$ for all the data sets, and selected the results of $N = 50$ since they generally allowed us to achieve higher performance ratios than $N = 8$ and $N = 25$, and did not significantly differ from $N = 100$. For all the methods, we used 5 linguistic terms per variable. Fuzzy-UCS was configured as detailed in Section IV-B.

The second group gathered a large number of learners with different knowledge representations: ZeroR, C4.5, IBk, Naïve Bayes, Part, SMO, GAssist, and UCS. ZeroR is a simple classifier system that always predicts the majority class in the training data set. We use this algorithm to provide a baseline result. C4.5 [11] is one of the most used decision trees, which derives from ID3 and introduces methods to deal with continuous variables and missing values. IBk [13] is a nearest neighbor algorithm; it decides the output of a new example as the most numerous class of the k nearest neighbors. Naïve Bayes [70] is a probabilistic classifier that estimates the parameters of a Bayesian model. Part [71] is a learning architecture that combines the creation of rules from partial decision trees and the separate-and-conquer rule learning technique to create a classifier without using global optimization. SMO [12] is a support vector machine that implements the *Sequential Minimization Algorithm*. GAssist [72] is a recent Pittsburgh-style LCS. UCS [5] is a Michigan-style LCS derived from XCS [3], [4] and specialized for supervised learning tasks. All the methods except for GAssist and UCS were run using Weka [73]. For GAssist, we used the open source code provided in [74]. For UCS, we used our own code. If not stated differently, all open source methods were configured with the parameters values recommended by default. For UCS we set (see [4], [5], [75] for notation details): $\mathrm{numIter} = 100{,}000$, $N = 6400$, $\mathrm{acc}_0 = 0.99$, $\nu = 10$, $\{\theta_{GA}, \theta_{\mathrm{del}}, \theta_{\mathrm{sub}}\} = 50$, $\chi = 0.8$, $\mu = 0.04$, $\delta = 0.1$, $r_0 = 0.6$. Fuzzy-UCS was configured with standard values as indicated in the previous section.

We applied the following statistical analysis to the results. We used the nonparametric Friedman's test [54], [55] to check whether all the learning algorithms performed the same on average. If significant differences were found, two procedures were applied to detect differences between methods. We first aimed at comparing the performance obtained by each of the inference types of Fuzzy-UCS to all other learners (instead of comparing all learners with the others as done in Section IV). To achieve this, we applied the nonparametric Bonferroni-Dunn [62] test. We computed this test as proposed in [52], where the critical value is calculated using the same equation as for the Nemenyi test [see (11)] but adjusting the critical values q_α according to the number of comparisons made (i.e., $n_\ell - 1$). Moreover, the analysis is complemented by performing pairwise comparisons among learners by means of a Wilcoxon signed-ranks test [58].

B. Comparison to Fuzzy Rule-Based Classification Systems

In the following, we compare the test performance and the interpretability of Fuzzy-UCS with the three types of inference to the aforementioned set of fuzzy rule-based learners.

Comparison of the performance. Table V details the test accuracies obtained with selected fuzzy learners and fuzzy UCS with the three inference schemes. The average performance of AdaBoost and MaxLogitBoost for the problems *ann* and *aud* is not provided since neither system was able to extract competent fuzzy rules from the two domains, leaving nearly all the feature space uncovered. The authors confirmed that this behavior could be due to high number of nominal attributes that these two problems have. The last two rows of the table provide the average rank and the absolute position in the ranking of each learner.

The experimental results show that the three configurations of Fuzzy-UCS were the best ranked in the comparison. The next methods in the ranking were the boosting algorithm Fuzzy LogitBoost, the genetic programming-based systems Fuzzy-GP and Fuzzy-SAP, and Fuzzy AdaBoost. Finally, we have Fuzzy GAP, and Fuzzy MaxLogitBoost.

Friedman's test rejected the hypothesis that all the methods performed the same on average at a significance level of 0.0001. Thence, we compared Fuzzy-UCS with each inference type with all the other learners to detect significant differences. Fig. 6 graphically represents the rank of each learner and groups the classifiers that perform equivalently to (1) Fuzzy-UCS with weighted average inference, (2) Fuzzy-UCS with action winner inference, and (3) Fuzzy-UCS with most numerous and fittest rules inference according to a Bonferroni-Dunn test at a significance level of 0.1. The statistical procedure supported the following hypotheses:

- Using Fuzzy-UCS with weighted average inference as the control learner, the statistical test supported the hypothesis that the performance of the control learner was equivalent to the performance of Fuzzy-UCS with the other two inference types. Moreover, Fuzzy-UCS with weighted average outperformed all the other learners.
- Using Fuzzy-UCS with action winner inference as the control learner, the test indicated that this learner performed equivalently to Fuzzy-UCS with the other two types of inference and Fuzzy LogitBoost.
- With respect to Fuzzy-UCS with most numerous and fittest rules inference, the test did not reject the hypothesis that all the fuzzy learners except for Fuzzy MaxLogitBoost and Fuzzy GAP performed equivalently on average.

We complemented the statistical study by comparing each pair of learners. Table VI shows the approximate p-values for

TABLE V
COMPARISON OF THE PERFORMANCE OF FUZZY-UCS WITH WEIGHTED AVERAGE (WAVG), ACTION WINNER (AWIN), AND MOST NUMEROUS AND FITTEST RULES (NFIT) WITH THE PERFORMANCE OF THE FUZZY LEARNERS

	GP	GAP	SAP	AdaBoost	LogitBoost	MaxLogitBoost	Fuzzy-UCS		
							wavg	awin	nfit
ann	77.86	77.20	78.02	-	76.20	-	98.85	97.39	98.61
aut	44.65	45.21	41.00	-	32.63	-	74.42	67.42	69.32
bal	69.73	64.33	65.80	85.54	88.30	75.58	88.65	84.40	83.40
bpa	56.62	57.91	62.30	65.34	64.46	56.53	59.82	59.42	58.93
cmc	47.00	46.57	46.27	49.55	51.10	45.21	51.72	49.67	49.42
col	79.15	73.51	81.89	63.06	63.06	63.06	85.01	82.46	78.50
gls	48.89	47.24	46.42	62.52	68.18	62.18	60.65	57.21	57.43
h-c	73.98	75.09	74.18	60.40	62.09	57.48	84.39	82.62	82.05
h-s	73.70	72.00	72.07	57.56	59.33	57.33	81.33	80.78	78.11
irs	94.47	90.80	91.53	95.47	95.33	92.00	95.67	95.47	93.73
pim	75.32	76.62	77.92	70.69	71.84	72.54	74.88	74.11	74.32
son	64.52	65.99	68.70	46.62	53.38	46.62	80.78	73.71	71.66
tao	80.36	81.75	81.15	91.46	91.73	84.52	81.71	83.02	87.53
thy	86.98	84.94	85.55	97.35	97.08	95.33	88.18	89.49	91.25
veh	46.16	44.59	42.96	30.82	37.25	38.05	67.68	65.35	65.34
wbcd	93.31	92.53	92.72	94.88	94.12	91.83	96.01	95.73	95.29
wdbc	90.93	90.49	91.52	37.26	62.74	37.26	95.20	94.61	94.51
wne	82.91	78.23	79.85	85.59	85.02	77.68	94.12	94.86	91.82
wpbc	74.77	74.47	74.37	23.65	76.35	23.65	76.06	76.05	71.69
zoo	71.18	66.65	66.08	41.89	41.89	41.89	96.50	94.78	95.90
Rank	*5.55*	*6.25*	*5.80*	*5.80*	*4.95*	*7.48*	*2.10*	*3.18*	*3.90*
Pos.	*5*	*8*	*6.5*	*6.5*	*4*	*9*	*1*	*2*	*3*

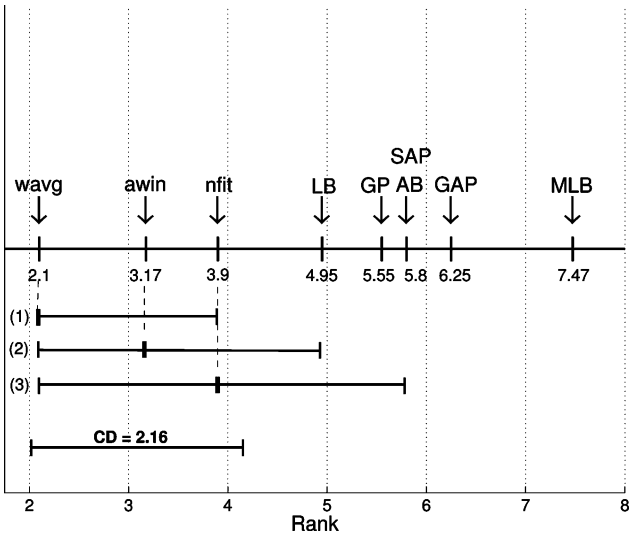


Fig. 6. Comparisons of one learner against the others with the Bonferroni-Dunn test at a significance level of 0.1. All the learners are compared to three different control groups: (1) Fuzzy-UCS with weighted average inference, (2) Fuzzy-UCS with action winner inference, and (3) Fuzzy-UCS with most numerous and fittest rules inference. The learners connected are those that perform equivalently to the control learner.

the pairwise comparison according to a Wilcoxon signed-ranks test. The symbols $\oplus$ and $\ominus$ indicate that the method in the row significantly improves/degrades the performance obtained with the method in the column. Similarly, the symbols $+$ and $-$ denote a nonsignificant improvement/degradation. The symbol $=$ indicates that each method outperforms and degrades the other in the same number of data sets. At a significance level of 0.05, the test indicated that Fuzzy-UCS with weighted average inference significantly outperformed all the other learners, including the two other types of inference of Fuzzy-UCS. Moreover, Fuzzy-UCS with action winner and most numerous and fittest inference schemes significantly improved all the other fuzzy learners.

Comparison of the interpretability. The study conducted in Section IV-B showed the interpretability-performance trade-off among the different inference schemes in Fuzzy-UCS. As shown, the excellent results of Fuzzy-UCS with weighted average with respect to all the other learners were hampered by the large number of fuzzy rules evolved by the method. The other two types of inference appeared as a positive alternative since they resulted in a moderate number of rules. Although they sightly degraded the accuracy rate with respect to the former approach, they were still valuable since they outperformed all the other fuzzy learners. The comparison made above confirmed the suitability of Fuzzy-UCS with action winner and most numerous and fittest rules inferences, since both systems significantly outperformed all the other fuzzy learners. In this section, we qualitatively analyze if the rule set evolved by these two methods is competitive in terms of readability.

TABLE VI
PAIRWISE COMPARISON OF THE PERFORMANCE OF FUZZY LEARNERS BY MEANS OF A WILCOXON SIGNED-RANKS TEST

	GP	GAP	SAP	AdaBoost	LogitBoost	MaxLogitBoost	Fuzzy-UCS		
							wavg	awin	nfit
GP		.0366	.2627	.0522	.4115	.0090	.0001	.0001	.0006
GAP	$\ominus$		.2180	.1005	.4781	.0187	.0002	.0002	.0002
SAP	$-$	$+$		.0674	.4330	.0111	.0003	.0005	.0036
AdaBoost	$-$	$-$	$-$		.0038	.0231	.0045	.0079	.0137
LogitBoost	$=$	$=$	$=$	$\oplus$		.0003	.0100	.0276	.0438
MaxLogitBoost	$\ominus$	$\ominus$	$\ominus$	$\ominus$	$\ominus$		.0005	.0009	.0007
Fuzzy-UCS (wavg)	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$		.0032	.0051
Fuzzy-UCS (awin)	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\ominus$		.2627
Fuzzy-UCS (nfit)	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\ominus$	$-$	

if y is triangle(-6.0,-3.0,0.0) **then** red
if ((x is trapezoid(3.0, 6.0) **or** x is trapezoid(3.0, 6.0))
or (x is triangle(0, 3.0, 6.0) **or** x is trapezoid(3.0, 6.0)))
and
((x is triangle (-3, 0.0, 3.0) **or** x is trapezoid(3.0, 6.0)
or . . .)
...
then blue

(a) GP-based learners

if x is L **and** y is L **then** blue **with** -5.42 **and** red **with** 0.0
if x is M **and** y is XS **then** blue **with** 2.21 **and** red **with** 0.0
if x is M **and** y is XL **then** blue **with** -2.25 **and** red **with** 0.0
...

(b) Boosting learners

if x is XL **then** blue **with** w=1.00
if x is XS **then** red **with** w=1.00
if x is {XS or S} **and** y is {XS or S} **then** red **with** w=0.87
...

(c) Fuzzy-UCS

Fig. 7. Examples of part of the models evolved by (a) the GP-based methods, i.e., Fuzzy GP, Fuzzy GAP, and Fuzzy SAP; (b) the boosting learners, i.e., Fuzzy AdaBoost, Fuzzy LogitBoost, and Fuzzy MaxLogitBoost; and (c) Fuzzy-UCS for the two-dimensional tao problem. In the fuzzy learners, we used the following five linguistic terms per variable: $\{XS, S, M, L, XL\}$. All fuzzy learners use triangular-shaped membership functions. Moreover, GP-based learners also use trapezoid-shaped membership functions.

As the type of rules evolved by the systems differ, we qualitatively evaluated the size of the models by extracting some characteristics. Fig. 7 shows examples of partial models evolved by the fuzzy learners for the tao problem. The models built by Fuzzy GP, Fuzzy GAP, and Fuzzy SAP consisted of a rule for each class of the domain. Each rule was directly extracted from an expression codified in a tree. The rules were represented by an arbitrary number of conjunctions (AND) and disjunctions (OR) of conditions over the variables of the domain. One example of these types of rules for a two-dimensional problem is

$$\begin{aligned}&\textbf{IF } \left(x_1 \textbf{ is } \widetilde{A_1^1} \textbf{ AND } x_2 \textbf{ is } \widetilde{A_2^1}\right)\\ &\textbf{OR } \left(x_1 \textbf{ is } \widetilde{A_1^2} \textbf{ AND } x_2 \textbf{ is } \widetilde{A_2^2}\right) \textbf{ THEN } c1\end{aligned} \tag{12}$$

where each variable x_i was represented by a linguistic term $\widetilde{A_i} = \{A_{i1} \vee \cdots \vee A_{\text{in}_i}\}$. All variables shared the same semantics which were defined by the combination of triangular-shaped and trapezoidal-shaped fuzzy membership functions [see Fig. 7(a)].

The fuzzy rule-based boosting algorithms created a set of linguistic fuzzy rules that take the following form:

$$\begin{aligned}&\textbf{IF } x_1 \text{ is } \widetilde{A_1} \text{ and } \cdots \text{ and } x_n \text{ is } \widetilde{A_n}\\ &\textbf{THEN } c_1 \textbf{ WITH } w_1^k, \ldots, c_m \textbf{ WITH } w_m^k\end{aligned} \tag{13}$$

where each variable x_i was represented by a linguistic term $\widetilde{A_i} = \{A_{i1} \vee \cdots \vee A_{\text{in}_i}\}$. All variables shared the same semantics, which was defined by means of triangular-shaped fuzzy membership functions [see Fig. 7(b)]. These boosting algorithms supported the absence of a variable by not assigning any linguistic term to the variable. The maximum population size N was a configuration parameter. In our experiments, the set $N = 50$.

To compare these two types of representations to the rule sets evolved by Fuzzy-UCS, we evaluated the size of the models as follows:

- We calculated the size of the models built by Fuzzy GP, Fuzzy GAP, and Fuzzy SAP by counting the number of AND, OR, and IS of the model. This gives us an idea of the average size of the rule. However, note that, due to the flexibility of these types of rules, it was not possible to directly compare them with the rules evolved by the three boosting algorithms and Fuzzy UCS. The rules evolved by Fuzzy GP, Fuzzy GAP, and Fuzzy SAP permit the combination of different logic operators, whose associativity and priority is given by the position of the operators in the tree. An equivalent conjunctive normal form for these rules could be found by applying De Morgan's laws. However, this transformation is not in the scope of this paper, and so, we only qualitatively evaluated the model sizes.

TABLE VII
SIZE OF THE MODELS EVOLVED BY THE FUZZY LEARNERS

	GP			GAP			SAP			AdaBoost	LBoost	MaxLBoost	Fuzzy-UCS		
	and	or	is	and	or	is	and	or	is				wavg	awin	nfit
ann	30.0	34.4	64.3	27.4	31.4	58.8	5.0	6.8	11.8	-	17.9	-	1038.6	27.2	12.7
aut	27.3	31.8	59.1	30.3	35.5	65.8	5.3	6.9	12.1	-	39.2	-	1555.7	45.1	28.7
bal	27.5	32.8	60.3	21.0	20.2	41.1	4.1	4.5	8.6	19.8	23.8	14.3	578.0	54.3	39.0
bpa	17.6	37.3	54.9	17.9	25.2	43.1	1.8	2.9	4.6	35.3	36.0	13.3	795.5	40.0	19.9
cmc	22.2	25.1	47.2	17.6	18.3	35.9	4.8	3.5	8.3	29.1	27.9	2.0	984.6	223.1	135.8
col	14.6	23.8	38.4	12.2	15.1	27.3	2.1	2.5	4.6	45.0	44.1	0.8	1469.3	50.8	26.0
gls	28.8	32.9	61.7	27.5	29.2	56.7	7.8	7.4	15.2	29.2	31.0	22.8	1293.7	27.8	14.5
h-c	13.6	20.0	33.6	11.4	13.5	24.9	1.7	2.6	4.3	39.6	38.9	1.0	1188.3	34.2	14.2
h-s	16.8	26.1	42.9	8.7	13.2	21.9	2.4	3.4	5.9	40.3	39.2	15.0	1173.0	37.2	18.8
irs	18.7	21.6	40.4	11.5	12.3	23.7	2.5	2.7	5.2	23.4	26.9	4.0	231.2	7.6	2.8
pim	18.3	19.9	38.2	13.7	14.6	28.3	2.0	1.7	3.7	36.9	34.0	13.3	1327.1	86.8	28.0
son	18.2	28.3	46.5	15.1	17.1	32.2	2.0	2.3	4.3	22.3	21.6	0.9	1208.1	70.7	63.4
tao	19.0	20.3	39.2	13.3	19.1	32.4	3.3	3.4	6.7	40.1	43.2	18.0	75.3	12.1	8.6
thy	18.2	20.0	38.1	12.4	13.0	25.4	2.8	2.4	5.1	25.7	29.4	8.8	624.4	16.3	5.0
veh	18.8	21.7	40.4	16.9	18.9	35.8	3.4	4.1	7.4	40.3	37.3	25.6	1641.7	143.8	63.7
wbcd	20.4	40.1	60.5	17.9	20.2	38.1	2.6	3.9	6.5	24.0	27.7	13.1	1033.9	38.9	8.4
wdbc	14.7	15.7	30.4	10.2	12.3	22.5	1.9	2.0	3.9	44.9	43.9	0.9	2108.7	105.4	38.4
wne	15.8	19.5	35.3	14.5	14.9	29.4	2.5	2.7	5.2	30.8	31.2	26.9	1437.7	33.1	9.0
wpbc	24.0	38.1	62.1	11.9	19.2	31.1	2.8	4.6	7.4	44.9	44.0	0.8	1536.6	62.3	45.3
zoo	34.0	34.9	68.9	37.8	38.2	76.0	8.2	9.7	17.9	42.0	36.2	0.9	263.4	5.0	3.3

- The size of the rule sets created by the boosting algorithms and Fuzzy-UCS were computed as:

$$\text{size} = \sum_{i=1}^{N} \frac{1}{\ell} \sum_{j=1}^{\ell} \frac{\text{maxLabels} - \text{numLabels}(x_i)}{\text{maxLabels} - 1} \tag{14}$$

  where N is the number of rules in the population, ℓ the number of variables, and maxLabels the number of linguistic labels (in our experiments, $\text{maxLabels} = 5$). This formula reckons the total number of variables in the model that have, at least, one linguistic term assigned. It also benefits general variables that have more than one linguistic label. To achieve a totally fair comparison, we also referred to the number of rules evolved by Fuzzy-UCS (see Table II).

Table VII shows the size of the models created by each fuzzy learner. Table VIII illustrates the approximate p-values resulting from the pairwise comparison between the learners according to a Wilcoxon signed-ranks test. For the three methods based on genetic programming, we considered the average number of variables for each rule (i.e., column is divided by the number of classes of the problems). The comparison shows that Fuzzy SAP, followed by Fuzzy GP, Fuzzy GAP, and Fuzzy MaxLogitBoost, were the methods that created the smallest models according to a Wilcoxon signed-ranks test at a significance level of 0.05. We have already discussed how the representation of Fuzzy GP, Fuzzy GAP, and Fuzzy SAP was much more flexible and by far less interpretable than the representation of the other learners (see the number of conjunctions and disjunctions with different associativity and priority in the rules). Thus, although the number of attributes per rule was smaller, the interpretability of the model was poor due to the flexibility of the rule [see the partial example provided for the tao problem in Fig. 7(a)]. Fuzzy-UCS with weighted average and with action winner created the biggest and the second biggest populations of the comparison. On the other hand, Fuzzy-UCS with most numerous and fittest rules created rule sets that, on average, were not significantly bigger than the rule sets built by Fuzzy-GP, Fuzzy AdaBoost, and Fuzzy LogitBoost; thus, disregarding the three learners based on genetic programming, whose rule sets were poorly readable due to the rule form, only Fuzzy MaxLogitBoost created more reduced populations.

The results provided in this section highlighted the high competitiveness of Fuzzy-UCS in terms of performance and interpretability with respect to other fuzzy learners. In Section V-C, we broaden the analysis and compare Fuzzy-UCS to a set of general purpose nonfuzzy learners.

C. Comparison With Nonfuzzy Learners

Now we compare Fuzzy-UCS to a set of general-purpose learners that use different knowledge representations: ZeroR, C4.5, IBk, Part, Naïve Bayes, SMO with polynomial kernels of order 3, SMO with Gaussian kernels, GAssist, and UCS. The systems were configured as recommended in the open source implementation, with exception of the following aspects. We ran IBk with $k = \{1, 3, 5\}$. We ranked the performance obtained by the three configurations, and we only provide the results with the settings that maximized the average rank, that is, $k = 5$ (IB5).

TABLE VIII
PAIRWISE COMPARISONS OF THE SIZES OF THE FUZZY LEARNERS BY MEANS OF A WILCOXON SIGNED-RANKS TEST

	GP	GAP	SAP	AdaBoost	LogitBoost	MaxLogitBoost	Fuzzy-UCS		
							wavg	awin	nfit
GP		.0003	.0001	.0005	.0001	.0137	.0001	.0003	.2179
GAP	⊖		.0001	.0002	.0001	.1671	.0001	.0002	.0228
SAP	⊖	⊖		.0001	.0001	.0276	.0001	.0001	.0001
AdaBoost	⊕	⊕	⊕		.8666	.0001	.0001	.0793	.1790
LogitBoost	⊕	⊕	⊕	−		.0001	.0001	.1005	.1084
MaxLogitBoost	⊖	−	⊕	⊖	⊖		.0001	.0002	.0105
Fuzzy-UCS (wavg)	⊕	⊕	⊕	⊕	⊕	⊕		.0001	.0001
Fuzzy-UCS (awin)	⊕	⊕	⊕	+	+	⊕	⊖		.0001
Fuzzy-UCS (nfit)	=	⊕	⊕	−	−	⊕	⊖	⊖	

TABLE IX
COMPARISON OF THE PERFORMANCE OF FUZZY-UCS WITH WEIGHTED AVERAGE (WAVG), ACTION WINNER (AWIN), AND MOST NUMEROUS AND FITTEST RULES INFERENCE (NFIT) WITH THE PERFORMANCE OF THE NON-FUZZY LEARNERS

	ZeroR	C4.5	IB5	Part	NaïveBayes	SMOp3	SMOrbf	GAssist	UCS	Fuzzy-UCS		
										wavg	awin	nfit
ann	76.20	98.90	97.34	98.57	86.33	99.34	91.90	97.88	99.05	98.85	97.39	98.61
aut	32.63	80.94	64.03	74.41	58.79	78.09	45.55	68.63	77.41	74.42	67.42	69.32
bal	45.46	77.42	88.18	82.86	90.57	91.20	88.30	79.57	77.32	88.65	84.40	83.40
bpa	57.99	62.31	58.85	67.56	55.97	59.97	57.99	62.24	67.59	59.82	59.42	58.93
cmc	42.70	52.62	46.51	50.04	50.65	48.75	42.70	53.58	50.27	51.72	49.67	49.42
col	63.06	85.32	81.49	84.51	78.23	75.59	82.41	94.30	96.26	85.01	82.46	78.50
gls	35.65	66.15	64.68	66.62	48.95	66.15	35.65	65.06	70.04	60.65	57.21	57.43
h-c	54.45	78.45	83.16	74.20	82.80	78.59	82.48	80.09	79.72	84.39	82.62	82.05
h-s	55.56	79.26	80.74	80.00	83.33	78.89	82.59	77.67	74.63	81.33	80.78	78.11
irs	33.33	94.00	96.00	94.00	96.00	92.67	93.33	96.20	95.40	95.67	95.47	93.73
pim	65.11	74.23	73.32	74.88	75.80	76.70	65.11	73.76	74.61	74.88	74.11	74.32
son	53.38	71.07	84.05	74.38	69.71	85.52	69.26	75.81	76.49	80.78	73.71	71.66
tao	49.89	95.92	97.14	94.33	80.98	84.22	83.63	91.59	87.00	81.71	83.02	87.53
thy	69.83	94.91	94.85	94.33	97.16	88.91	69.83	92.52	95.13	88.18	89.49	91.25
veh	25.42	71.14	68.91	73.39	46.28	83.30	41.71	67.00	71.40	67.68	65.35	65.34
wbcd	65.52	94.99	97.14	95.71	96.15	96.42	96.13	95.59	96.28	96.01	95.73	95.29
wdbc	63.11	94.40	96.78	94.46	93.13	97.58	92.88	94.24	95.96	95.20	94.61	94.51
wne	39.93	93.89	96.67	93.30	97.19	97.75	39.93	93.19	96.13	94.12	94.86	91.82
wpbc	72.97	71.61	78.85	70.05	69.45	81.25	72.97	72.33	69.40	76.06	76.05	71.69
zoo	41.89	92.81	90.47	93.81	94.47	97.83	76.03	93.97	96.78	96.50	94.78	95.90
Rank	*11.50*	*5.95*	*5.28*	*5.95*	*6.63*	*4.28*	*8.95*	*6.25*	*4.65*	*4.68*	*6.50*	*7.40*
Pos	*12*	*5.5*	*4*	*5.5*	*9*	*1*	*11*	*7*	*2*	*3*	*8*	*10*

The analogous process was carried out for SMO with polynomial kernels. We experimented with polynomial kernels of order 1 and 3, and supplied the results obtained with polynomial kernels of order 3 since they maximized the average rank. We did not introduce the same system with different configurations in the comparison to avoid biasing the statistical analysis of the results.

Comparison of the performance. Table IX shows the accuracy of the learners on the same collection of real-world problems. The two last rows of the table provide the average rank and the position in the ranking of each learner.

Several observations can be drawn from the results. First, let us highlight the good performance presented by Fuzzy-UCS with weighted average inference. This learner is the third best method in the ranking. Its average rank is really close to UCS, by which Fuzzy-UCS was inspired. Thus, the fuzzy representation does seem not to limit the capabilities of Fuzzy-UCS if all the evolved rules are used to infer the class of new examples. Moreover, the average rank is also close to the best ranked method: SMO with polynomial kernels. The other two inference schemes presented higher average ranks. Fuzzy-UCS with action winner

TABLE X
PAIRWISE COMPARISON OF THE PERFORMANCE OF NON-FUZZY LEARNERS BY MEANS OF A WILCOXON SIGNED-RANKS TEST

	ZeroR	C4.5	IB5	Part	NaïveBayes	SMOp3	SMOrbf	GAssist	UCS	Fuzzy-UCS		
										wavg	awin	nfit
ZeroR		.0001	.0001	.0001	.0001	.0001	.0010	.0001	.0001	.0001	.0001	.0001
C4.5	$\oplus$		.7089	.9039	.2627	.4209	.0072	.9405	.2043	.7938	.3905	.0793
IB5	$\oplus$	=		.7938	.0534	.4115	.0004	.4553	.8519	.8228	.1084	.0304
Part	$\oplus$	+	−		.2471	.2959	.0057	.4330	.2180	.4209	.3135	.0251
NaïveBayes	$\oplus$	−	−	−		.0674	.0333	.1084	.1354	.0333	.1790	.3703
SMOp3	$\oplus$	+	+	+	+		.0032	.2959	.6542	.1672	.0400	.0366
SMOrbf	$\oplus$	$\ominus$	$\ominus$	$\ominus$	$\ominus$	$\ominus$		.0032	.0064	.0004	.0025	.0152
GAssist	$\oplus$	−	−	−	+	−	$\oplus$		.2180	.5016	.3135	.0859
UCS	$\oplus$	+	=	+	+	−	$\oplus$	+		.4330	.0674	.0366
Fuzzy-UCS (wavg)	$\oplus$	+	=	+	$\oplus$	−	$\oplus$	+	−		.0032	.0051
Fuzzy-UCS (awin)	$\oplus$	=	−	−	+	$\ominus$	$\oplus$	−	−	$\ominus$		.2627
Fuzzy-UCS (nfit)	$\oplus$	−	$\ominus$	$\ominus$	+	$\ominus$	$\oplus$	−	$\ominus$	$\ominus$	−	

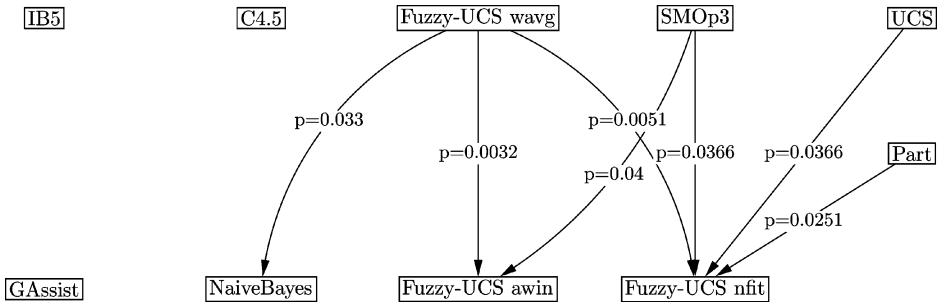


Fig. 8. Illustration of the significant differences (at $\alpha = 0.05$) of performance among non-fuzzy methods and Fuzzy-UCS. An edge $L_1 \xrightarrow{p_{\text{value}}} L_2$ indicates that the learner L_1 outperforms the learner L_2 with the corresponding p_{value}. To facilitate the visualization, ZeroR and SMO with Gaussian kernels, the two most outperformed algorithms, were not included in the graph.

inference and most numerous and fittest rules inference occupy the seventh and ninth position in the ranking.

Multiple-comparison Friedman's test rejected the hypothesis that all the learners performed the same on average at a significance level of 0.0001. Post-hoc Bonferroni-Dunn test only permitted to reject the hypothesis that the best ranked learners performed equivalently to Fuzzy-UCS with the most numerous and fittest rules inference, SMO with Gaussian kernel, and Zero-R. However, the test has a low discriminatory power for large numbers of learners [52]. Thus, we also compared the performance of each pair of learners by means of a Wilcoxon signed-ranks test (see Table X). Fig. 8 uses a graph to illustrate the significant differences between learners. The test confirmed that Fuzzy-UCS with weighted average inference was one of the best learners in the comparison. It significantly outperformed Naïve Bayes, SMO with Gaussian kernels, ZeroR, and Fuzzy-UCS with the other two types of inference. Moreover, Fuzzy-UCS with weighted average inference did not significantly degrade the results obtained with any other learner. Fuzzy-UCS with action winner inference was only significantly outperformed by SMO with polynomial kernels, and Fuzzy-UCS with weighted average inference. Besides, it significantly improved SMO with Gaussian kernel and ZeroR. Fuzzy-UCS with most numerous and fittest rules inference presented the poorest results among the three configurations of Fuzzy-UCS. It significantly degraded the results obtained by SMO with polynomial kernels, UCS, IB5, Part, and Fuzzy-UCS with weighted average inference. However, note that it performed equivalently to well-known algorithms such as C4.5, Naïve Bayes, and GAssist.

Comparison of the interpretability. Here, we qualitatively compare the readability of the models created by the different learners. We do not consider IBk, SMO, and Naïve Bayes since their knowledge representation can hardly be compared to the other learners. IBk is a lazy classifier that does not use any knowledge representation; to predict the output of a new input example, IBk returns the majority class among its k nearest neighbors. SMO represents the knowledge by $\binom{n_c}{2}$ support vector machines (where n_c is the number of classes), each one consisting of a set of real-valued weights. Therefore, the models created by these two learners are very difficult to interpret. On the other hand, Naïve Bayes builds interpretable models formed by a set of parameters which estimate the independent probability functions and the so-called class-prior of a Bayesian model. In [76], a close connection between Naïve Bayes and Neuro-Fuzzy Classifier Systems was identified, providing a framework that maps a Naïve Bayes classifier into a Neuro-Fuzzy Classifier with the aim of improving its capabilities. The discussion on the difference in the interpretability of Naïve Bayes and their similarity to Neuro-Fuzzy Classifier

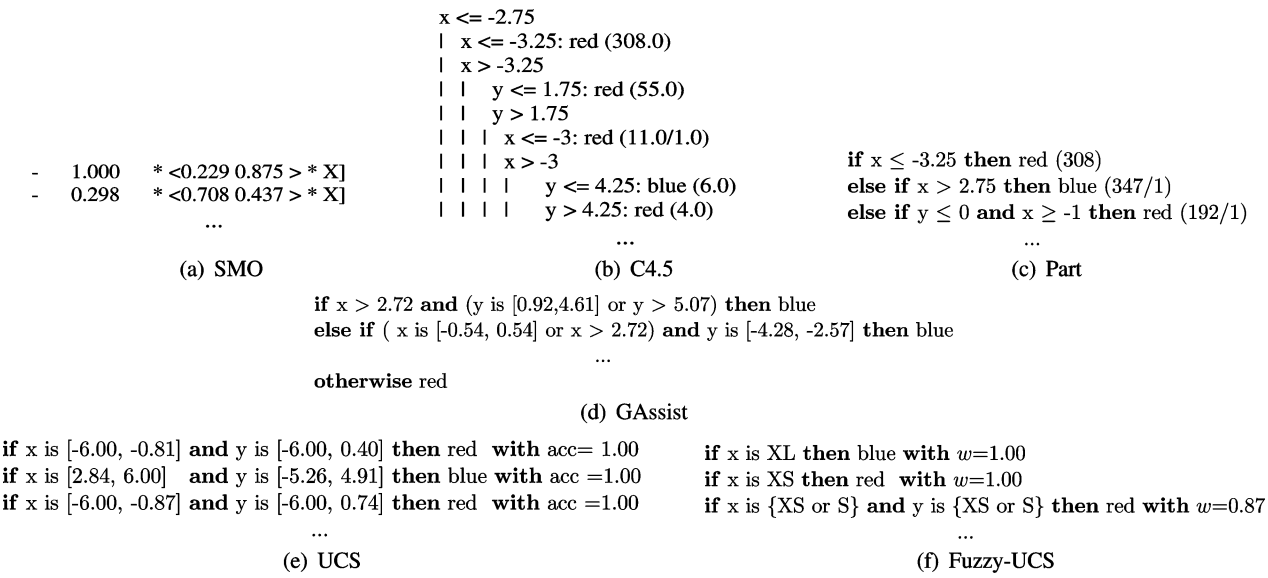


Fig. 9. Examples of part of the models evolved by (a) SMO, (b) C4.5, (c) Part, (d) GAssist, (e) UCS, and (f) Fuzzy-UCS for the two-dimensional tao problem.

Systems or Fuzzy Rule-Based Systems is out of the scope of this paper. The reader is referred to [76] for further details.

Thus, in the remainder of this analysis we focus on the comparison of the rule-based and tree-based learners, i.e., C4.5, Part, GAssist, UCS, and Fuzzy-UCS. Fig. 9 plots examples of the models evolved by these learners for the two-dimensional tao problem; besides, an example of the weights created by SMO is also depicted. C4.5 evolves trees in which the nodes represent a decision over one variable [see Fig. 9(b)]. We evaluated the model size by counting the number of leaves of the tree. Part and GAssist create a set of rules which are defined by conjunction of conditions over their variables, and are interpreted as an ordered activation list [see Fig. 9(c) and (d)]. Moreover, GAssist uses a default rule. UCS evolves a rule set similar to Fuzzy-UCS, but replacing linguistic rules with interval-based rules [see Fig. 9(e)]. Each rule can be regarded as an expert classifier in the region of the feature space it covers. We used the number of rules evolved as the metric of interpretability for Part, GAssist, UCS, and Fuzzy-UCS, although we acknowledge that the measure is not directly comparable as we later discuss. Note that we did not use (14) to compute the model size because some of the learners are represented by an ordered activation list.

Table XI shows the model sizes of the rule-based and tree-based systems. Qualitatively, it is worth mentioning the following aspects.

- Fuzzy-UCS with weighted average, jointly with UCS, were the two methods in the ranking with higher performance from those that use a rule-based representation. Thus, when performance prevails over interpretability, Fuzzy-UCS is a good approach to face new problems.
- Fuzzy-UCS with weighted average inference, as well as the other two inference schemes, significantly created smaller populations than UCS according to a Wilcoxon signed-ranks test (at $\alpha = 0.05$). Thus, Fuzzy-UCS achieved one of the main objectives of this work: to create smaller models than those evolved by UCS.

TABLE XI
AVERAGE SIZES OF THE MODELS BUILD BY C4.5, PART, GASSIST, UCS, AND FUZZY-UCS WITH WEIGHTED AVERAGE (WAVG), ACTION WINNER (AWIN), AND MOST NUMEROUS AND FITTEST RULES INFERENCE (NFIT)

	C4.5	Part	GAssist	UCS	Fuzzy-UCS		
					wavg	awin	nfit
ann	38	15	5	4494	2769	75	36
aut	44	21	7	4565	3872	114	74
bal	45	37	8	2177	1212	114	75
bpa	25	9	6	2961	1440	73	39
cmc	162	168	15	3634	1881	430	271
col	5	9	5	3486	4135	154	81
gls	24	15	5	3359	2799	62	36
h-c	29	21	6	2977	3574	113	46
h-s	17	18	5	3735	3415	117	62
irs	5	4	3	1039	480	18	7
pim	19	7	7	3605	2841	192	62
son	14	8	5	520	3042	178	160
tao	36	17	6	807	111	19	14
thy	8	4	4	1994	1283	37	11
veh	69	32	7	4941	3732	332	147
wbcd	12	10	3	2334	3130	138	28
wdbc	11	7	4	5206	5412	276	101
wne	5	5	3	3685	3686	95	26
wpbc	12	7	4	5299	3772	156	115
zoo	11	8	6	1291	773	16	10

- Fuzzy-UCS was the only method in the comparison in which the same semantics (adapted to the universe of discourse of each variable) is shared among all variables, and only 5 linguistic terms were specified. Consequently, Fuzzy-UCS rules were more readable.

The results also indicate that, even the moderate-sized populations provided by Fuzzy-UCS with action winner inference and

most numerous and fittest rules, these techniques still are not competitive (in number of rules) to Part and C4.5 (if we consider the number of leaves as a comparative measure to the number of rules), and especially to GAssist. However, two important distinctions need to be considered to justify these differences.

- Fuzzy-UCS and, in general, Michigan-style LCSs evolve rules that, by themselves, are experts on the region of the feature space that they cover and collaborate to classify all the input space. Thus, each rule can be regarded as an expert classifier; if the human expert is only interested in a region of the feature space, only the rules involved in this region need to be considered. On the other hand, the rules evolved by Part and GAssist form an ordered activation list. That is, to classify a new example, rules are checked in order and the first rule that matches determines the output. In the case of GAssist, a default rule is used to classify all the examples not matched by any rule in the activation list. This implies that all the previous rules need to be considered to understand why the system is giving this prediction.
- Fuzzy-UCS evolves the rule set incrementally, whilst the other learners go through the data several times to extract the classification model. Incremental learning gives a big advantage to Fuzzy-UCS when learning from large data sets.

The analysis supplied in this section showed that Fuzzy-UCS is highly competitive with respect to a large set of general-purpose machine learning techniques. The proposed weighted average version of Fuzzy-UCS was one of the best performers. Thus, a fuzzy rule-based system could achieve accuracy rates as good as—or even better than—other machine learning techniques with knowledge representations that have poor meaning for human experts such as support vector machines or instance based algorithms. Moreover, Fuzzy-UCS with the two other inference schemes appeared also to be competitive. Fuzzy-UCS with action winner inference evolved substantially reduced rule sets, although not as much as the ones evolved by GAssist and Part, and it was only statistically surpassed by SMO with polynomial kernels, and our Fuzzy-UCS with weighed average inference. Section VI explores the capabilities of Fuzzy-UCS to learn from large volumes of data.

VI. Fuzzy-UCS for Mining Large Data Sets

The two essential differences between Fuzzy-UCS and other rule-based learners are that Fuzzy-UCS a) does not perform any form of global optimization, and b) incrementally evolves the rule-based knowledge. Based on a rule set roughly initialized in the first learning iterations by the covering operator, the system is responsible for incrementally evaluating the parameters of the rules and refining the rule-based knowledge by creating more general and more accurate rules. This process provides two main advantages with respect to other learners.

- Fuzzy-UCS learns from a stream of examples. This enables the system to learn from changing environments. This differs from other machine learning methods, such as C4.5, IBk, SMO, and Pittsburgh-style LCSs, which need to process all the training data set in order to produce the final model.
- The learning can be stalled whenever required, and the evolved rule set can be used for predicting the class of new input examples. The more learning iterations the system has performed, the more general and accurate the rules should be. Consequently, the cost of the algorithm increases linearly with the maximum population size N, the number of variables per rule a, and the number of learning iterations n_{learn}

$$\text{Cost}_{\text{Fuzzy-UCS}} = O\left(a \cdot N \cdot n_{\text{learn}}\right) \tag{15}$$

but it does not depend directly on the number of examples. In static data sets, it is recommended that n_{learn} be, at least, the number of examples in the training data set.

In this section, we exploit the benefits of on-line learning in Fuzzy-UCS and apply the system to mine very large data sets. Specifically, we test the performance of Fuzzy-UCS on the 1999 KDD Cup intrusion detection data set [77]. In the following, we describe the data set and present the experimental results.

A. Data Set Description

The 1999 KDD Cup intrusion detection data set gathers a large collection of examples of a wide variety of network intrusions simulated in a military environment. We used the subset of 494 022 examples provided in [77] that advocate 23 different classes. Examples consist of 35 continuous attributes and 6 nominal attributes, which usually characterize network traffic behavior. We used a ten-fold cross validation procedure to estimate test accuracy.

B. Results

We ran Fuzzy-UCS on the KDD'99 domain with the default configuration as in the previous section except for $\theta_{\text{GA}} = 200$ and $P_{\#} = 0.2$. We increased the period of GA application $(\theta_{\text{GA}} = 200)$ to permit the classifiers to receive more parameter updates before undergoing a genetic event. We also diminished the probability of generalization in covering $(P_{\#} = 0.2)$ since the dispersion of the examples was very low. We ran the experiment for 2 000 000 learning iterations, so that Fuzzy-UCS only received each learning instance an average of 5 times.

Fig. 10 plots the evolution of the test performance and the population size of Fuzzy-UCS with action winner inference in the first 150 000 learning iterations. Note that the system quickly evolved a highly accurate population. After seeing the first 35 000 examples, that is, a 7% of the whole training data set, the test performance was already about 99%. Increasing the number of learning iterations did not significantly improve the average performance, but it did create more general and equally accurate classifiers. This behavior can be observed in Table XII, which depicts the test accuracy and the rule set size obtained by Fuzzy-UCS with weighted average inference (first column), action winner inference (second column), and most numerous and fittest rules inference (third column) at different learning iterations. That is, every 500 000 learning iterations, we used the corresponding test set to calculate the accuracy with the three types of inference. While sampling the test examples, all

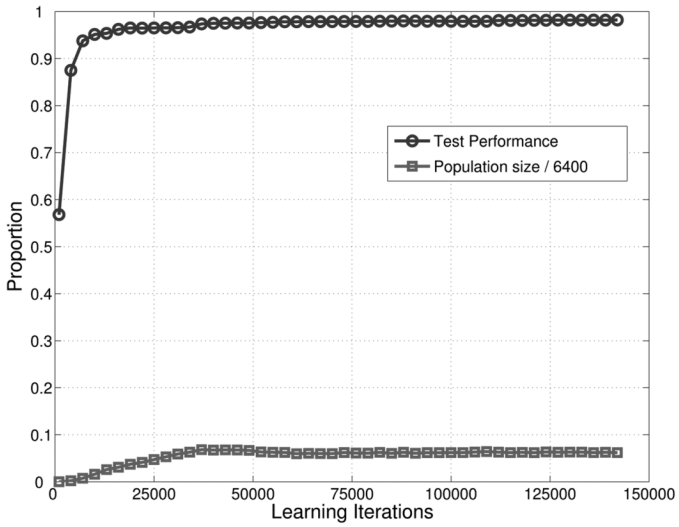


Fig. 10. Evolution of test accuracies and the population size of Fuzzy-UCS with action winner inference in the first 150 000 learning iterations of the 1999 KDD Cup data set.

TABLE XII
TEST PERFORMANCE AND NUMBER OF RULES EVOLVED BY FUZZY-UCS IN THE 1999 KDD CUP INTRUSION DETECTION DATASET AT DIFFERENT NUMBERS OF LEARNING ITERATIONS

#Iter	wavg		awin		nfit	
	perf.	#rules	perf.	#rules	perf.	#rules
500,000	99.32	1944	99.13	541	99.27	417
1,000,000	99.36	2089	99.07	492	99.25	369
1,500,000	99.37	2178	99.02	460	99.24	350
2,000,000	99.36	2257	99.00	428	99.19	323

the learning mechanisms of Fuzzy-UCS were disabled, so that rule set was not modified.

The results show that the number of rules in the final population for the action winner and the most numerous and fittest rules inference decreased as the number of learning iterations increased. Thus, the system was pushing the population to obtain maximally general and accurate rules. This behavior was not so clear with the weighted average inference since this inference scheme used all the experienced rules in the final population that have positive fitness, regardless of their generality.

Finally, let us highlight the differences of Fuzzy-UCS with respect to a Pittsburgh-style LCS. In the last section, we configured GAssist with a population of 400 individuals. At the initialization phase, these 400 individuals needed to be evaluated. Thus, the condition of all the rules of each classifier was matched with each of the training instances. This means that, in the population initialization, a Pittsburgh-style LCSs would go through all the data set 400 times, seeing about 180 000 000 instances. The use of windowing mechanisms, as the ones implemented in GAssist, would permit us to reduce the number of instances that each individual matches to be evaluated by a constant value; nevertheless, note that, in any case, the number of evaluations would increase linearly with the number of input examples. After that, the system would have only the first approximation, and the evolutionary pressures would create new individuals that needed to be evaluated. This makes these types of systems computationally expensive for large data sets. On the other hand, note that Fuzzy-UCS only needed to see 35 000 examples to extract a highly accurate model, and that further iterations were to create a more general rule set. These results emphasize the advantages of on-line learning.

VII. SUMMARY, SELF-ANALYSIS, AND FURTHER WORK

A. Summary

In this paper, we proposed a Michigan-style on-line Learning Fuzzy-Classifier System for supervised learning which iteratively evolves a set of linguistic fuzzy rules which collaborate to cover all the input space. Three schemes of inference and reduction algorithms were designed to infer the output of unknown examples from reduced rule sets. These three mechanisms were proposed to offer different levels of rule set reduction and consequently lead to different accuracy rates.

We performed a detailed analysis of the performance and interpretability of the rule sets evolved by Fuzzy-UCS. First, we carefully analyzed the three inference and reduction mechanisms in Fuzzy-UCS. Second, we also compared Fuzzy-UCS with six fuzzy-rule-based learners and nine general-purpose learners with different types of representation. The analysis showed that Fuzzy-UCS was highly competitive to both groups of learners. The many benefits of the on-line fuzzy-rule-based architecture, as well as some drawbacks detected in this study, are detailed in the SWOT analysis of Section VII-B.

In the final step of the analysis, we exploited the incremental learning architecture of Fuzzy-UCS to extract a model from a large data set: the 1999 KDD Cup intrusion detection data set. It was found that Fuzzy-UCS could quickly evolve a highly accurate model, having only seen the ten percent of the total number of examples in the domain. Incremental learning enabled us to have a rough approximation of the model after a few thousand learning iterations, further refining the rule set as the system received more examples.

B. SWOT Analysis

All the evidence provided through the experimentations is summarized in the SWOT analysis presented in Table XIII, where *strengths* represent the main advantages of Fuzzy-UCS, *weaknesses* show its drawbacks, *opportunities* outline some suggested further lines of investigation, and *threats* include some optional approaches considered by other methods that could compete with our proposal.

Fuzzy-UCS has four main strengths. First, the system presented a high performance, which supports the use of Fuzzy-UCS in complex problems. Second, it uses linguistic fuzzy rules, which are much more readable than interval-based rules since all the variables share the same semantics and only a small number of linguistic terms per variable are defined (specifically, in our experiments we only used five linguistic terms per variable). This is really important for domains with high dimensionality where each variable presents different ranges. Third, Fuzzy-UCS is an on-line process that performs incremental learning, and so, the system neither has knowledge about the data set nor does any kind of global optimization.

TABLE XIII
SWOT ANALYSIS OF FUZZY-UCS

STRENGTHS	WEAKNESSES
- It shows a high performance regarding error rate; comparable with the state-of-the-art in classification - It uses a highly legible representation based on linguistic fuzzy rules - It performs incremental, on-line learning - It is capable of mining large data sets	- It generates moderate or large sized rule sets (depending on the configuration) - Although it can deal with real, integer or categorical features, only application for real and integer data types is recommended
OPPORTUNITIES	**THREATS**
- It may be applied in data streams; further analysis will be conducted - Because of the use of fuzzy logic, the algorithm could be adapted to deal with vague and uncertain data - The proposed seminar system opens opportunities for further research on fuzzy knowledge representation, the fuzzy inference engine, and evolutionary operators	- A small number of interval-based rules can be interpreted more easily than a large number of linguistic fuzzy rules - Other learning approaches combined with preprocessing can also deal with large data sets

And fourth, since the run-time complexity of Fuzzy-UCS does not depend on the number of instances in the data set, our system is very useful for mining large amounts of data as we showed with the KDD'99 problem, which consists of about half a million instances, 41 features, and 23 classes.

The main weakness of the system is that, despite the application of reduction schemes, it evolves slightly larger rule sets than those created by other machine learning techniques such as GAssist. Consequently, the number of rules may hinder the interpretability of the evolved knowledge. However, it is worth highlighting the comments made in Section V-C about the type of rules evolved by GAssist and Fuzzy-UCS, which may approach their readability capacities. GAssist does not share any semantics between variables, makes the rules available in an ordered activation list, and uses a default rule. A less important feature of our system is that, although it can work with categorical input variables, fuzzy rules are especially useful for real or integer-valued variables, since in the former case the rule would be equivalent to a classical crisp (or nonfuzzy) one.

We also want to honestly mention some possible threats to Fuzzy-UCS. On the one hand, an expert might find a small number of interval-based rules more legible than many linguistic fuzzy rules (in fact, the degree of interpretability of a system is very difficult to assess when different knowledge representations are compared). On the other hand, there are hybrid learning approaches to deal with problems with large data sets, such as the inclusion of preprocessing algorithms to reduce the data set size, which would allow some of the systems compared in this paper to address these problems.

Finally, the proposed Fuzzy-UCS algorithm shows some interesting opportunities which will be developed in future work. First, because of its incremental learning capability, the system can be applied to extract information from data streams, which is currently a topic of increasing interest [78]. Second, the use of fuzzy logic allows the system to be adapted for managing vague and uncertain data, very common in many real-world problems [79]. Furthermore, as future work we can consider the inclusion of some of the techniques proposed by other systems (such as inference based on an activation list with default rule as in GAssist) and the design of new techniques to achieve greater reductions of the fuzzy rule set without a significant loss of test performance, as well as a more detailed research of other fuzzy knowledge representations.

REFERENCES

[1] J. H. Holland, "Adaptation," in *Progress in Theoretical Biology*, R. Rosen and F. Snell, Eds. New York: Academic, 1976, vol. 4, pp. 263–293.
[2] J. Holland and J. Reitman, "Cognitive systems based on adaptive algorithms," in *Pattern-Directed Inference Systems*, D. Waterman and F. Hayes-Roth, Eds. San Diego, CA: Academic, 1978, pp. 313–329.
[3] S. W. Wilson, "Classifier fitness based on accuracy," *Evol. Comput.*, vol. 3, no. 2, pp. 149–175, 1995.
[4] S. W. Wilson, "Generalization in the XCS classifier system," in *Proc. 3rd Annu. Conf. Genetic Programming*, 1998, pp. 665–674.
[5] E. Bernadó-Mansilla and J. M. Garrell, "Accuracy-Based learning classifier systems: Models, analysis and applications to classification tasks," *Evol. Comput.*, vol. 11, no. 3, pp. 209–238, Sep. 2003.
[6] M. V. Butz, D. E. Goldberg, and P. L. Lanzi, "Gradient descent methods in learning classifier systems: Improving XCS performance in multistep problems," *IEEE Trans. Evol. Comput.*, vol. 9, no. 5, pp. 452–473, Oct. 2005.
[7] M. V. Butz, P. L. Lanzi, and S. W. Wilson, "Function approximation with XCS: Hyperellipsoidal conditions, recursive least squares, and compaction," *IEEE Trans. Evol. Comput.*, vol. 12, no. 3, pp. 355–376, Jun. 2008.
[8] J. Bacardit and M. V. Butz, "Data mining in learning classifier systems: Comparing XCS with gassist," in *Proc. 7th Int. Workshop Learning Classifier Syst.*, 2004.
[9] M. V. Butz, *Rule-Based Evolutionary Online Learning Systems: A Principled Approach to LCS Analysis and Design*, ser. Studies in Fuzziness and Soft Computing. New York: Springer, 2006, vol. 109.
[10] A. Orriols-Puig and E. Bernadó-Mansilla, "Evolutionary rule-based systems for imbalanced datasets," *Soft Comput. J.*, 2008, 10.1007/s00500-008-0319-7.
[11] J. R. Quinlan, *C4.5: Programs For Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1995.
[12] J. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods—Support Vector Learning*. Cambridge, MA: MIT Press, 1998.
[13] D. W. Aha, D. F. Kibler, and M. K. Albert, "Instance-based learning algorithms," *Mach. Learn.*, vol. 6, no. 1, pp. 37–66, 1991.

[14] L. Bull, M. Studley, A. Bagnall, and I. Whittley, "Learning classifier system ensembles with rule-sharing," *IEEE Trans. Evol. Comput.*, vol. 11, no. 4, pp. 496–502, Aug. 2007.

[15] E. Bernadó-Mansilla and T. K. Ho, "Domain of competence of XCS classifier system in complexity measurement space," *IEEE Trans. Evol. Comput.*, vol. 9, no. 1, pp. 1–23, Feb. 2005.

[16] S. W. Wilson, "Compact rulesets from XCSI," in *Advances in Learning Classifier Systems*. New York: Springer, 2002, pp. 197–210.

[17] C. Fu and L. Davis, "A modified classifier system compaction algorithm," in *GECCO'02: Proc. 2002 Genetic Evol. Comput. Conf.*, San Francisco, CA, 2002, pp. 920–925.

[18] P. W. Dixon, D. W. Corne, and M. J. Oates, "A ruleset reduction algorithm for the XCSI learning classifier system," in *Learning Classifier Systems*. New York: Springer, 2004, vol. 2661/2003, pp. 20–29.

[19] O. Cordón, F. Herrera, F. Hoffmann, and L. Magdalena, *Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases*, ser. Advances in Fuzzy Systems—Aplications and Theory. Singapore: World Scientific, 2001, vol. 19.

[20] M. Valenzuela-Radón, "The fuzzy classifier system: A classifier system for continuously varying variables," in *Proc. 4th ICGA*. San Mateo, CA: Morgan Kaufmann, 1991, pp. 346–353.

[21] M. Valenzuela-Radón, "Reinforcement learning in the fuzzy classifier system," *Expert Syst. Appl.*, vol. 14, pp. 237–247, 1998.

[22] A. Parodi and P. Bonelli, "A new approach to fuzzy classifier systems," in *Proc. 5th Int. Conf. Genetic Algorithms*, 1993, pp. 223–230.

[23] T. Furuhashi, K. Nakaoka, and Y. Uchikawa, "Suppression of excess fuzziness using multiple fuzzy classifier systems," in *Proc. 3th IEEE Int. Conf. Fuzzy Syst.*, 1994, pp. 411–414.

[24] J. Velasco, "Genetic-based on-line learning for fuzzy process control," *Int. J. Intell. Syst.*, vol. 13, pp. 891–903, 1998.

[25] H. Ishibuchi, T. Nakashima, and T. Murata, "Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 29, no. 5, pp. 601–618, Oct. 1999.

[26] J. Casillas, B. Carse, and L. Bull, "Fuzzy-XCS: A michigan genetic fuzzy system," *IEEE Trans. Fuzzy Syst.*, vol. 15, no. 4, pp. 536–550, Aug. 2007.

[27] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: Univ. Michigan Press, 1975.

[28] D. E. Goldberg, *Genetic Algorithms in Search, Optimization & Machine Learning*, 1st ed. New York: Addison-Wesley, 1989.

[29] S. F. Smith, "A learning system based on genetic adaptive algorithms," Ph.D., Univ. Pittsburgh, Pittsburgh, PA, 1980.

[30] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[31] J. S. Aguilar-Ruiz, R. Giraldez, and J. C. Riquelme, "Natural encoding for evolutionary supervised learning," *IEEE Trans. Evol. Comput.*, vol. 11, no. 4, pp. 466–479, Aug. 2007.

[32] T. Kovacs, "Towards a theory of strong overgeneral classifiers," in *Foundations of Genetic Algorithms, Volume 6*. San Mateo, CA: Morgan Kaufmann, 2000, pp. 165–184.

[33] A. Orriols-Puig and E. Bernadó-Mansilla, "Revisiting UCS: Description, Fitness Sharing and Comparison With XCS," in *Advances at the Frontier of LCSs*. New York: Springer.

[34] K. Nakaoka, T. Furuhashi, and Y. Uchikawa, "A study on apportionment of credits of fuzzy classifier system for knowledge acquisition in large scale systems," in *Proc. 3th IEEE Int. Conf. Fuzzy Syst.*, 1994, pp. 1797–1800.

[35] H. Ishibuchi, T. Yamamoto, and T. Murata, "Hybridization of fuzzy gbml approaches for pattern classification problems," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 35, no. 2, pp. 359–365, Apr. 2005.

[36] A. Bonarini, "Evolutionary learning of fuzzy rules: Competition and cooperation," in *Fuzzy Modelling: Paradigms and Practice*. Norwell, MA: Kluwer, 1996, pp. 265–284.

[37] A. Bonarini and V. Trianni, "Learning fuzzy classifier systems for multi-agent coordination," *Inf. Sci. Int. J.*, vol. 136, no. 1-4, pp. 215–239, 2001.

[38] L. Bull and J. Hurst, "ZCS redux," *Evol. Comput.*, vol. 10, no. 2, pp. 185–205, 2002.

[39] M. Butz, T. Kovacs, P. L. Lanzi, and S. W. Wilson, "Toward a theory of generalization and learning in XCS," *IEEE Trans. Evol. Comput.*, vol. 8, no. 1, pp. 28–46, Feb. 2004.

[40] M. V. Butz, D. E. Goldberg, P. L. Lanzi, and K. Sastry, "Problem solution sustenance in XCS: Markov chain analysis of niche support distributions and the impact on computational complexity," *Genet. Progr. Evolv. Mach.*, vol. 8, no. 1, pp. 5–37, 2007.

[41] O. Cordón, M. J. d. Jesús, and F. Herrera, "A proposal on reasoning methods in fuzzy rule-based classification systems," *Int. J. Approx. Reason.*, vol. 20, no. 1, pp. 21–45, 1999.

[42] H. Ishibuchi and T. Yamamoto, "Rule weight specification in fuzzy rule-based classification systems," *IEEE Trans. Fuzzy Syst.*, vol. 13, no. 4, pp. 428–435, Aug. 2005.

[43] D. E. Goldberg, *The Design of Innovation: Lessons from and For Competent Genetic Algorithms*. Norwell, MA: Kluwer, 2002.

[44] H. Ishibuchi, T. Nakashima, and T. Morisawa, "Voting in fuzzy rule-based systems for pattern classification problems," *Fuzzy Sets Syst.*, vol. 103, no. 2, pp. 223–238, 1999.

[45] D. Nauck and R. Kurse, "How the learning of rule weights affects the interpretability of fuzzy systems," in *Proc. 1998 IEEE Int. Conf. Fuzzy Syst.*, Anchorage, AK, 1998, vol. 2, pp. 1235–1240.

[46] H. Ishibuchi and T. Nakashima, "Effect of rule weights in fuzzy rule-based classification systems," *IEEE Trans. Fuzzy Syst.*, vol. 9, no. 4, pp. 506–515, Aug. 2001.

[47] M. J. d. Jesus, F. Hoffmann, L. J. Navascués, and L. Sánchez, "Induction of fuzzy-rule-based classifiers with evolutionary boosting algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 12, no. 3, pp. 296–308, Jun. 2004.

[48] J. Otero and L. Sánchez, "Induction of descriptive fuzzy classifiers with the logitboost algorithm," *Soft Comput.*, vol. 10, no. 9, pp. 825–835, 2006.

[49] E. Bernadó-Mansilla, X. Llorà, and J. M. Garrell, "XCS and gale: A comparative study of two learning classifier systems on data mining," in *Advances in Learning Classifier Systems*, ser. LNAI. New York: Springer, 2002, vol. 2321, pp. 115–132.

[50] A. Asuncion and D. J. Newman, UCI machine learning repository 2007 [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

[51] T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," *Neural. Comput.*, vol. 10, no. 7, pp. 1895–1924, 1998.

[52] J. Demsar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, 2006.

[53] D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*. London, U.K.: Chapman & Hall, 2000.

[54] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *J. Amer. Statist. Assoc.*, vol. 32, pp. 675–701, 1937.

[55] M. Friedman, "A comparison of alternative tests of significance for the problem of m rankings," *Ann. Math. Statist.*, vol. 11, pp. 86–92, 1940.

[56] R. A. Fisher, *Statistical Methods and Scientific Inference*, 2nd ed. New York: Hafner, 1959.

[57] P. B. Nemenyi, "Distribution-Free multiple comparisons," Ph.D., Princeton Univ., Princeton, NJ, 1963.

[58] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics*, vol. 1, pp. 80–83, 1945.

[59] S. W. Wilson, "Get real! XCS with continuous-valued inputs," in *Learning Classifier Systems. from Foundations to Applications*, ser. LNAI. Berlin, Germany: Springer-Verlag, 2000, pp. 209–219.

[60] M. V. Butz, T. Kovacs, P. L. Lanzi, and S. W. Wilson, "How XCS evolves accurate classifiers," in *Proc. 2001 Genetic Evol. Comput. Conf.*, San Francisco, CA, 2001, pp. 927–934.

[61] G. Brown, T. Kovacs, and J. A. R. Marshall, "UCSpv: Principled voting in UCS rule populations," in *Proc. 2007 Conf. Genetic Evol. Comput.*, New York, 2007, pp. 1774–1781.

[62] O. J. Dunn, "Multiple comparisons among means," *J. Amer. Statist. Assoc.*, vol. 56, pp. 52–64, 1961.

[63] L. Sánchez and I. Couso, "Learning with imprecise examples with GA-P algorithms," *Soft Comput.*, vol. 5, no. 1–4, pp. 305–319, 1998.

[64] L. Sánchez and I. Couso, *Fuzzy Random Variables-Based Modeling with GA-P Algorithms*. Norwell, MA: Kluwer, 2000, pp. 245–256.

[65] L. Sánchez, I. Couso, and J. A. Corrales, "Combining GP operators with SA search to evolve fuzzy rule based classifiers," *Inform. Sci.*, vol. 136, no. 1–4, pp. 175–191, 2001.

[66] J. Korst and E. Aarts, *Simulated Annealing and Boltzmann Machines*. New York: Wiley, 1997.

[67] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Int. Conf. Mach. Learn.*, 1996, pp. 148–156.

[68] L. Sánchez and J. Otero, "Boosting fuzzy rules in classification problems under single-winner inference: Research articles," *Int. J. Intell. Syst.*, vol. 22, no. 9, pp. 1021–1034, 2007.

[69] J. Alcalá-Fdez, L. Sánchez, S. García, M. J. d. Jesus, S. Ventura, J. M. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas, J. C. Fernández, and F. Herrera, "Keel: A software tool to assess evolutionary algorithms to data mining problems," *Soft Comput.*, to be published.
[70] G. H. John and P. Langley, "Estimating continuous distributions in Bayesian classifiers," in *Proc. 11th Conf. Uncertainty Artificial Intell.*, 1995, pp. 338–345.
[71] E. Frank and I. H. Witten, "Generating accurate rule sets without global optimization," in *Proc. 15th Int. Conf. Mach. Learning*, San Francisco, CA, 1998, pp. 144–151.
[72] J. Bacardit, "Pittsburgh genetic-based machine learning in the data mining ERA: Representations, generalization and run-time," Ph.D. dissertation, Ramon Llull Univ., Barcelona, Spain, 2004.
[73] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. San Francisco, CA: Morgan Kaufmann, 2005.
[74] J. Bacardit, GAssist source code [Online]. Available: http://www.asap.cs.nott.ac.uk/jqb/PSP/GAssist-Java.tar.gz, 2007
[75] M. V. Butz and S. W. Wilson, "An algorithmic description of XCS," in *Adv. Learning Classifier Systems: Proc. 3rd Int. Workshop*, 2001, vol. 1996, pp. 253–272.
[76] A. Nurnberger, C. Borgelt, and A. Klose, "Improving naive bayes classifiers using neuro-fuzzy learning," in *Proc. 1999 Conf. Neural Inf. Process.*, Perth, Australia, 1999, vol. 1, pp. 154–159.
[77] S. Hettich and S. D. Bay, The UCI KDD archive University of California, Department of Information and Computer Science, Irvine, CA, 1999 [Online]. Available: http://kdd.ics.uci.edu
[78] C. Aggarwal, *Data Streams: Models and Algorithms*, C. Aggarwal, Ed. New York: Springer, 2007.
[79] L. Sánchez and I. Couso, "Advocating the use of imprecisely observed data in genetic fuzzy systems," *IEEE Trans. Fuzzy Syst.*, vol. 15, no. 4, pp. 551–562, Aug. 2007.

Albert Orriols-Puig received the M.Sc. graduate degree in computer science in 2004 from the Universitat Ramon Llull, Barcelona, Spain. He is currently pursuing the Ph.D. degree at the Universitat Ramon Llull and is a member of the research group "Grup de Recerca en Sistemes Intel·ligents." His dissertation focuses on a detailed analysis and enhancement of learning classifier systems to further apply these learning techniques to challenging real-life problems.

In 2006, he was a Visiting Researcher at the Illinois Genetic Algorithm Laboratory (IlliGAL), University of Illinois at Urbana-Champaign. Since 2007, he has been collaborating and visiting the Soft Computing and Intelligent Information Systems Research Group, University of Granada, Spain, where he has been working on the integration of fuzzy logic into learning classifier systems. His research interests include on-line evolutionary learning, fuzzy modeling, learning from rarities, data complexity, and machine learning in general.

Jorge Casillas received the M.Sc. and Ph.D. degrees in computer science in 1998 and 2001, respectively, from the University of Granada, Spain.

He is an Associate Professor with the Department of Computer Science and Artificial Intelligence, University of Granada, where he is a member of the Soft Computing and Intelligent Information Systems Research Group. He has been Visiting Research Fellow at the University of the West of England, Bristol, U.K. He has edited two international books, edited two international journal special issues, and organized four special sessions in international conferences on the topics "interpretability-accuracy trade-off in fuzzy modeling," "genetic fuzzy systems," and "intelligent robotics." He is author of more than 20 journal papers, 10 book chapters, and 40 conference papers. He is on the editorial board of the journal *Evolutionary Intelligence*. His research interests include fuzzy modeling, intelligent robotics, marketing intelligent systems, knowledge discovery, and metaheuristics.

Dr. Casillas is the treasurer of the European Society for Fuzzy Logic and Technologies (EUSFLAT) and coordinator of the working group on Genetic Fuzzy Systems.

Ester Bernadó-Mansilla received the B.Sc. degree in telecommunications engineering, the M.Sc. degree in electronic engineering, and the Ph.D. degree in computer science from the Enginyeria i Arquitectura La Salle, Universitat Ramon Llull, Barcelona, Spain, in 1992, 1995, and 2002, respectively.

During 2002, she was a Visiting Researcher at the Computing Sciences Research Center, Bell Laboratories, Lucent Technologies, Murray Hill, NJ. She is currently an Assistant Professor in the School of Computer Science and Engineering, Enginyeria i Arquitectura La Salle, Ramon Llull University, and the Director of the research group "Grup de Recerca en Sistemes Intel·ligents." Her research interests include genetic algorithms, machine learning, pattern recognition, and data mining.