A Comparative Study of Several Genetic-Based Supervised Learning Systems

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

[1] Enginyeria i Arquitectura La Salle, Universitat Ramon Llull, 08022, Barcelona, Spain, aorriols@salleURL.edu, esterb@salleURL.edu
[2] Department of Computer Science and Artificial Intelligence, University of Granada, 18071, Granada, Spain, casillas@decsai.ugr.es

Summary. This chapter gives insight in the use of Genetic-Based Machine Learning (GBML) for supervised tasks. Five GBML systems which represent different learning methodologies and knowledge representations in the GBML paradigm are selected for the analysis: UCS, GAssist, SLAVE, Fuzzy AdaBoost, and Fuzzy LogitBoost. UCS and GAssist are based on a non-fuzzy representation, while SLAVE, Fuzzy AdaBoost, and Fuzzy LogitBoost use a linguistic fuzzy representation. The models evolved by these five systems are compared in terms of performance and interpretability to the models created by six highly-used non-evolutionary learners. Experimental observations highlight the suitability of GBML systems for classification tasks. Moreover, the analysis points out which systems should be used depending on whether the user prefers to maximize the accuracy or the interpretability of the models.

1 Introduction

Genetic-Based Machine Learning (GBML) [30] gathers a wide range of learning techniques that use *Evolutionary Algorithms* (EAs) [5,26,30] for knowledge discovery. Research on this topic was historically conducted from two perspectives: Pittsburgh-style [12] and Michigan-style [30] GBML systems. Recently, the increase in the understanding of how evolutionary algorithms work [27] has propelled the research on GBML. As a result, the concept of GBML has been extended with new proposals of learners that use EAs to evolve their knowledge. Some of these approaches lay between the definitions of Pittsburgh-style and Michigan-style GBML, such as the *Iterative Learning Rule* approach [47]. Others methodologies propose to include EAs as robust search mechanisms to assist the building of neural networks [34, 48] or statistical classifiers [14, 36].

Among the different new approaches, evolutionary algorithms have been combined with *Fuzzy Rule-Based Systems*, resulting in the so-called *Genetic Fuzzy Rule-Based Systems* (GFRBSs) [11]. GFRBSs use EAs either to learn fuzzy rules or to tune different components of the fuzzy system. The main

A. Orriols-Puig et al.: *A Comparative Study of Several Genetic-Based Supervised Learning Systems*, Studies in Computational Intelligence (SCI) **125**, 205–230 (2008)
www.springerlink.com
© Springer-Verlag Berlin Heidelberg 2008

advantage of the fuzzy representation is that it allows for a better interpretability of the knowledge evolved, providing a flexible, robust, and powerful methodology to deal with *noisy*, *imprecise*, and *incomplete* data.

The aim of this work is to evaluate the performance and the interpretability of the models evolved by five different GBML architectures in data mining tasks, and to compare their behavior to other non-evolutionary techniques. We include representatives of the different tendencies of GBML for supervised learning in the comparison. We select two GBML systems that use a non-fuzzy (*crisp*) representation: UCS [6], and GAssist [4]; and three GBML methods based on a fuzzy representation: SLAVE [29], Fuzzy AdaBoost [14], and Fuzzy LogitBoost [36]. We compare these GBML systems to six highly-used non-evolutionary techniques. These learners come from different learning paradigms such as instance-based learning, rule and decision-tree induction, statistical modeling, and neural networks. The algorithms are compared on a collection of 20 real-world datasets extracted from the UCI repository [8] and local repositories [7].

The remaining of this paper is organized as follows. Section 2 presents the different approaches in GBML, and Sect. 3 briefly explains the five GBML systems selected for the comparison. Section 4 details the experimentation comparison and present the results. Finally, Sect. 5 summarizes and concludes the work.

2 Genetic-Based Machine Learning

Firstly designed regarding the animal behavior [30], research on GBML has been historically conducted from two different perspectives: the *Pittsburgh* approach [12], and the *Michigan* approach [30,31]. Recently, a third methodology has received an increasing amount of attention: the *Incremental Rule Learning* approach [47]. This three families are briefly introduced as follows.

Pittsburgh-Style GBML Systems follow the essence of genetic algorithms [27,30]. Each individual in the population consists of a rule set that covers all the input space. The quality of an individual is estimated considering different aspects such as its accuracy and size. The typical genetic operators, i.e., selection, crossover, and mutation, are adapted to deal with rule sets instead of binary strings. At the end of the learning process, Pittsburgh-style GBML systems return the best individual found during the search, which is used to predict the class of unknown examples. The first successful developments of Pittsburgh-style GBML for supervised learning are GABIL [13] and GIL [32]. A new generation Pittsburgh-style GBML derived from GABIL can be found in GAssist [4].

Michigan-Style GBML Methods are *cognitive systems* that combine a credit-apportionment system, usually adapted from a reinforcement learning method [44], and evolutionary algorithms to evolve a population of accurate rules.

Differently from the Pittsburgh approach, each individual is represented by a single rule, which is evaluated incrementally by the credit-apportionment system. An evolutionary algorithm is applied with a certain frequency on the population to discover new promising rules. At the end of the learning process, all the rules in the population contribute to decide the output of new test examples. Some of the first developments of Michigan-style GBML are SCS [26] and NewBoole [9]. Although these systems showed to be able to solve different classification tasks, some drawbacks were also detected, mainly associated to the achievement of accurate generalizations. This led to further research that culminated in the design of XCS [50, 51], by far the most influential Michigan-style GBML system. XCS works under a reinforcement learning scheme. The system can be used for classification tasks by considering a reinforcement learning problem in which maximum payoffs correspond to correct classifications and minimum payoffs to incorrect classifications. On the other hand, classification tasks can be solved in a more straightforward way using UCS [6], a system which inherits the main components from XCS but specializes them to supervised learning.

Iterative Rule Learning (IRL), firstly proposed in the context of the *SIA* system [47], uses a separate-and-conquer methodology to create an ordered list of individuals. Each individual is represented by a single rule, as in the Michigan approach. The system iteratively invokes an evolutionary algorithm, which evaluates the individuals according to their accuracy and generality. The best individual returned by the EA is added to the end of the ordered list of rules, and all the matching examples are removed from the training dataset. This process is repeated until the training dataset is empty. In test mode, the predicted class of a new example is given by the first rule in the ordered list that matches the example. One of the best representatives of IRL system is HIDER [1].

The big advances in the understanding of evolutionary algorithms and the proposal of the first *competent evolutionary algorithms* [27, 37, 38] has led to the hybridization of EAs and different machine learning techniques, resulting in a wide variety of new GBML systems. In the field of statistical learning, EAs have been applied to discover new promising rules in different boosting algorithms such as AdaBoost [21] and LogitBoost [22]. In the realm of neural networks, EAs have been used to evolve either the weights [34] or the structure of neural networks [48]. The flexibility provided by EAs has also been used to construct Fuzzy Rule-Based Systems (FRBSs), resulting in the so-called GFRBS [11]. Research on this field has mainly focused on the use of evolutionary algorithms to tune different components of an FRBS, such as the fuzzy sets or the fuzzy rules.

For the study performed in this paper, we selected five rule-based GBML systems that belong to different approaches and use either fuzzy or crisp knowledge representations. Specifically, we chose two of the most significant crisp GBML methods for supervised learning, which use intervalar

representations: (a) UCS [6], a Michigan-style GBML system, and (b) GAssist [4], a Pittsburgh-style GBML technique. We also selected three GBML algorithms that use a fuzzy-rule representation: (c) SLAVE [10, 28], an IRL algorithm, and the boosting techniques (d) Fuzzy AdaBoost [14] and (e) Fuzzy LogiBoost [36], two implementations of the AdaBoost and LogitBoost algorithms that use EAs to evolve fuzzy rules. Next section introduces these five learners.

3 Description the GBML Systems Used in the Comparison

This section briefly describes the five GBML systems included in the comparison: UCS [6], GAssist [4], SLAVE [29], Fuzzy AdaBoost [14], and Fuzzy LogitBoost [36]. The reader is referred to the original papers for further information about these methods.

3.1 UCS

UCS [6] is a Learning Classifier System which inherits the main components of XCS [50,51], but specializes them to be applied only to supervised learning tasks. In the following, we describe the main components of the system.

Knowledge Representation UCS evolves a population of classifiers which jointly cover the input space. Each classifier consists of a production rule of the form $condition \rightarrow class$ and a set of parameters. The condition specifies the set of inputs where the classifier can be applied. For continuous inputs, the condition is codified as a set of intervals $[l_i, u_i]^n$, which globally represents a hyperrectangle in the feature space. The class c^k of the rule specifies the class predicted when the condition is satisfied:

$$\textbf{if } x_1 \in [l_1, u_1] \wedge \ldots \wedge x_n \in [l_n, u_n] \textbf{ then } c^k \tag{1}$$

Each rule has the following parameters: a) accuracy *acc*; b) fitness F; c) correct set size *cs*; d) numerosity *num*; and e) experience *exp*. Accuracy and fitness are measures of the quality of the classifier. The correct set size is the estimated average size of all the correct sets where the classifier participates. Numerosity is the number of copies of the classifier, and experience is the number of times that a classifier has belonged to a match set.

Learning Interaction During training, UCS incrementally evolves a set of classifiers. At each learning iteration, the system receives an input example e and its class c. Then, the system creates the match set [M], which contains all the classifiers in the population [P] whose condition matches e. From that, the correct set [C] is formed, which consists of all the classifiers in [M] that

predict class c. If [C] is empty, the covering operator is activated, creating a new classifier with a generalized condition matching e, and predicting class c.

Next, the parameters of all the classifiers in [M] are updated. The experience of each classifier is increased, and its accuracy is updated depending on whether the current prediction was correct. The correct set size cs is calculated if the classifier belongs to [C]. Then, as proposed in [35], the fitness is shared among all the classifiers that participate in [C].

After one learning cycle, a genetic algorithm (GA) is triggered if the average time since the last application of the GA on the classifiers in [C] is greater than θ_{GA}. In this case, the GA selects two parents from [C] with a probability that depends on the classifier's fitness. The two parents are copied, creating two new children, which are recombined and mutated with probabilities χ and μ respectively. Recombination crosses the parent's conditions by two points. Mutation modifies the lower and upper bound of an interval according to a uniform distribution. Finally, each offspring is introduced into the population, removing another classifier if the population is full.

3.2 GAssist

GAssist [4] is one of the most competitive Pittsburgh-style GBML systems. GAssist was initially derived from GABIL [13], introducing several modifications that able the system to overcome scalability problems detected in the first Pittsburgh-style GBML approaches [20]. The rule representation and the learning interaction are described as follows.

Knowledge Representation

GAssist evolves a set of individuals, each of them represented by a variable-length set of rules:

$$Ind = (R_1 \vee R_2 \vee \ldots R_n) \tag{2}$$

where each rule consists of a condition and a predicted class c^k:

$$\textbf{IF } (x_1 = V_1^1 \vee \ldots \vee x_1 = V_m^1) \wedge \ldots \wedge (x_n = V_n^1 \wedge \ldots \wedge x_n = V_n^k) \textbf{ THEN } c^k \tag{3}$$

That is, each input variable x_i is represented by a disjunction of feasible values for this variable. For nominal variables, $(V_1^i ... V_j^i)$ represent the j possible values that the variable can take. For continuous variables, GAssist applies a discretization technique to transform the input space into intervals of values. Several discretization techniques have been proposed for GAssist. In our experiments, we used a uniform discretization.

Learning Interaction

The core of the system is a near-standard genetic algorithm similar to the one applied in GABIL [13]. Thus, at each learning iteration, the system selects a set of individuals, and applies crossover and mutation generating a new set of offspring. These offspring are evaluated by means of a fitness function based on the *minimum description length principle* (MDL) [41]. GAssist uses the same crossover operator defined by GABIL, i.e., a *semantically correct crossover* operator [13]. This is a multiple-point crossover operator that forces that the selected points cut both parents in the same position of the variable. The mutation operator randomly adds or removes one value of a given variable.

GAssist introduces a new *deletion operator* that permits to remove rules from individuals, and so, to control their size. This operator is activated after a predefined number of iterations, and it removes the rules of an individual that do not match any input example. To avoid an excessive loss of diversity, which may have a negative effect in subsequent recombinations, this operator is not applied if the individual does not have a minimum number of rules.

Finally, GAssist controls the runtime of the system by means of a windowing scheme addressed as *Incremental Learning with Alternating Strata* (ILAS). This mechanism splits the training dataset into several non-overlapping subsets of examples, and selects a different subset at each GA iteration. Thus, ILAS permits to reduce the training time of a single iteration of the GA since a lower number of examples need to be match with the new individuals in the evaluation process. Moreover, in [4], it was empirically shown that this technique allows for a better generalization.

3.3 SLAVE

SLAVE [10, 28] is an inductive learning algorithm based on a fuzzy-rule representation. The system follows an iterative rule learning scheme. In the following, the knowledge representation and the learning interaction are explained.

Knowledge Representation

SLAVE creates a set of individuals whose condition is represented in *conjunctive normal form*:

$$\textbf{IF } x_1 \text{ is } \widetilde{A_1^k} \wedge \ldots \wedge x_n \text{ is } \widetilde{A_n^k} \textbf{ THEN } c^k$$

where each input variable x_i is represented by a disjunction of linguistic terms $\widetilde{A_i^k} = \{A_{i1} \text{ or } ... \text{ or } A_{in_i}\}$, and the consequent c^k is the class predicted by the rule. In our experiments, all the variables share the same semantics, which are defined by means of triangular-shaped fuzzy membership functions. 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). Then, the matching degree of the rule is determined by the T-norm (conjunction) of the matching degree of all the input variables.

In the inference process, the class of a new example is determined by the rule that maximizes the matching degree with this example. In case of having more than one rule with maximum matching degree, the class of the rule with maximum fitness is selected as the output.

Learning Interaction

SLAVE iteratively evolves a set of individuals following an iterative rule learning scheme [47]. This process is based on the iteration of the following steps: (a) learn one rule from the dataset, (b) penalize the data covered by the rule.

Figure 1 illustrates the learning scheme of SLAVE. Given a data set E and a specific class B, the system searches for the rule that describes this class more accurately. This process is performed by a steady-state genetic algorithm. The fitness function of the GA is determined by the train error and the generality of the rule. Then, this rule is aggregated to the fuzzy-rule set. If more rules are required to represent all the examples of class B, the examples covered by the current rules of class B are removed, and the GA is run again providing a new rule for class B. The same procedure is repeated until no more rules are required for class B. Then, the same algorithm is followed to learn rules for the other classes of the domain, resulting in a rule set that covers all the instances in the training data set.

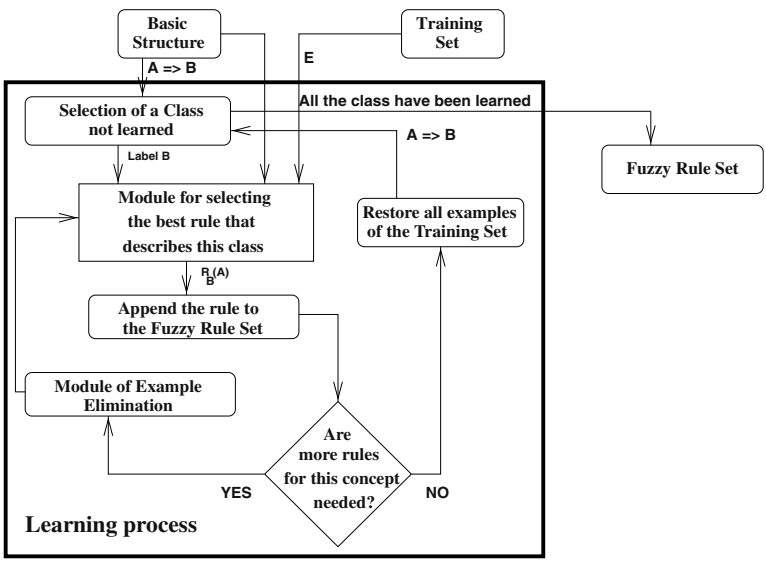

Fig. 1. Illustrative scheme of the learning process of SLAVE

3.4 Fuzzy AdaBoost

Fuzzy AdaBoost [14] is an evolutionary boosting algorithm that applies the AdaBoost algorithm [21] to learn a fuzzy-rule-based classifier. In the following, we introduce the knowledge representation and the learning procedure of Fuzzy AdaBoost.

Knowledge Representation

Fuzzy AdaBoost creates a set of *weak classifiers*, which take the following form:

$$\textbf{IF } x_1 \text{ is } A_1^k \wedge ... \wedge x_n \text{ is } A_n^k \textbf{ THEN } c_1^k \textbf{ WITH } s_1^k, \ ... \ , c_p^k \textbf{ WITH } s_p^k$$

Each input variable x_i is represented by a *linguistic term* A_i^k. All variables share the same semantics, represented by triangular-shaped membership functions. The method permits the absence of a variable by not assigning any linguistic term to this variable. In the consequent, the rule maintains one weight s_j^k for each class j that indicates the soundness with witch the weak classifier predicts the class j.

The fuzzy inference is as follows. Given an input instance e, it is assigned to the class c that maximizes the following expression:

$$\arg\max_{k=1,\cdot,p} \bigvee_{j=1}^{N} A^j(e) \wedge s_k^j \tag{4}$$

where N is the number of classifiers in the population. $\vee$ and $\wedge$ are the t-norm and the t-conorm operators respectively.

Learning Interaction

Fuzzy AdaBoost iteratively invokes an algorithm that provides a *low quality classifier*, addressed as *weak hypothesis* in the boosting literature. Each example i in the training dataset has associated a weight w^i, and the search for promising classifiers focuses on the examples that have higher weights. Each time that a new weak hypothesis is added to the compound classifier, the examples in the training dataset are re-weighted. In that way, in next iterations, the search will be focused toward examples that are more difficult to learn. Moreover, at the end of each iteration, a voting strength α is assigned to each weak hypothesis, which depends on the confidence in the classification accuracy of that rule. In the following, these three steps briefly explained.

1. *Creation of weak classifiers* Fuzzy AdaBoost uses a integer-coded genetic algorithm [26] to evolve a population of the aforementioned weak classifiers. The best weak classifier generated by the GA is added to the compound classifier. The fitness of each classifier is computed as a combination of

two objectives. The first objective considers the generality of the rule, i.e., the proportion of examples that matches with high degree. The second objective computes the accuracy of the rule. Thus, the GA pressures toward highly general and accurate rules.

2. *Computation of the strength of a classifier* Fuzzy AdaBoost updates the strength α^j of a classifier according to the error of the rule generated in the previous generation j. That is, low errors result in high values of α^j; that means that the rule will have a strong influence in the inference of the class.
3. *Update of the weights of the training examples* After one iteration of the learning algorithm (and so, the generation of a new rule), the weights w^i of the examples in the training dataset are updated. Specifically, the weight of a correctly classified instance w_i is decreased according to the matching degree of the new rule with this instance and the strength α^j of the rule. On the other hand, the weight of incorrectly classified instances is increased, so that new rules will focus on the classification of these instances.

Class Inference of Test Examples

Given a new unknown example e, Fuzzy AdaBoost predicts the output of e as follows. Each rule j votes for each class it predicts according to (4), resulting in a vote $g^c(e)$ for a class c. Then, the votes for each class are added:

$$\text{vote}_k = \sum_{j:class_j=k} \alpha^j \cdot g^j(e) \tag{5}$$

The most voted class is returned as output.

3.5 Fuzzy LogitBoost

The LogitBoost algorithm [22] is a boosting method similar to AdaBoost that uses a greedy version of generalized backfitting [22] to build an additive model. It has been experimentally shown that LogitBoost outperforms AdaBoost, especially in multi-class problems. Due to these improvements, LogitBoost was extended to induce fuzzy classifiers, resulting in the so-called Fuzzy LogitBoost algorithm [36]. This learning technique inherits the main components from Fuzzy AdaBoost – such as the knowledge representation, the learning scheme, and the voting scheme – , but introduces the corrections proposed by LogitBoost. In the following, we detail the learning interaction, and refer the reader to Sect. 3.4 for details on the knowledge representation and class inference of new input examples.

Learning Interaction

Fuzzy LogitBoost follows a learning scheme similar to AdaBoost. The goal of the algorithm is to minimize the likelihood of the compound classifier. For details on the statistical formulation of the problem, the reader is referred

to [22, 36, 42]. Instead, this section presents the final algorithm for Fuzzy LogitBoost.

Algorithm 3.1 shows the pseudocode for Fuzzy LogitBoost. The algorithm generates N weak classifiers, where N is a configuration parameter. The process to generate a new classifier is the following (see lines 4–22 of the algorithm). For each training example, the probability of the output class i, given the rule j and the example k is computed according to the function f_{ijk}, which represents the additive model. p_{ijk} is used to update the weigths w_{ijk}. Next, a genetic algorithm is applied to find a rule that fits accurately the training data, which is subject to the fitness function given in line 10. Then, the function values f_{ijk} are updated for each class i according to the best classifier found by the GA. Finally, the new rule is added to the compound classifier.

4 Experimentation

This section analyzes the competence of the five GBML systems in classification tasks. In pattern classification, we aim at obtaining accurate models which provide comprehensible explanations for human experts. For this purpose, we select a set of real-world problems and compare the performance and rule set interpretability of the GBML systems to a set of highly-competent and widely-used machine learning techniques. In the following, we first present the experimentation methodology and then compare the five GBML to the other learners.

Algorithm 3.1: Outline of the fuzzy logitboost algorithm

```
1  Algorithm: Logitboost
2  f_{i0k} = 0
3  for j = 1,···,N do
4      for k = 1,···,p do
5          for i = 1,···,n do
6              p_{ijk} = e^{f_{ij-1k}} / (1 + e^{f_{ij-1k}})
7              w_{ijk} = p_{ijk}(1 − p_{ijk})
8          end
9          Find rule antecedent A^j that minimizes fitness(A^j), where:
10             fitness(A^j) = Σ_i^n w_{ijk} ( s^j · A^j(x_i) − (y_{ik} − p_{ijk}) / w_{ijk} )^2
11         and
12             s^j = Σ_i (y_{ik} − p_{ijk}) A^j(x_i) / Σ_i w_{ijk} [A^j(x_i)]^2
13         for i = 1,···,n do
14             f_{ijk} = f_{ij-1k} + s^j · A^j(x_i)
15         end
16         Aggregate the new rule to the compound classifier
17     end
18 end
```

4.1 Methodology

Herein, we present (a) the real-world problems chosen for the experimentation, (b) the learning techniques included in the comparison, (c) the metrics used to evaluate the learners performance and models interpretability, and (d) the statistical analysis applied to evaluate the results.

Experimentation Problems

We selected a collection of 20 real-world problems with different characteristics (see Table 1) which may pose particular challenges to the different learning techniques. All these problems were obtained from the UCI repository [8], except for *tao*, which was chosen from a local repository [7].

Machine Learning Techniques Included in the Comparison

We compared the five GBML methods to C4.5, IBk, Naive Bayes, Part, SMO, and Zero-R. C4.5 [40] is a decision tree that enhances ID3 by introducing

Table 1. 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), and the proportion of instances with missing values (%MisInst)

Id.	Dataset	#Inst	#Fea	#Re	#In	#No	#Cl	%MisInst
ann	Annealing	898	38	6	0	32	5	0
aut	Automobile	205	25	15	0	10	6	22.4
bal	Balance	625	4	4	0	0	3	0
bpa	Bupa	345	6	6	0	0	2	0
cmc	Contraceptive method choice	1,473	9	2	0	7	3	0
col	Horse colic	368	22	7	0	15	2	98.1
gls	Glass	214	9	9	0	0	6	0
h-c	Heart-c	303	13	6	0	7	2	2.3
h-s	Heart-s	270	13	13	0	0	2	0
irs	Iris	150	4	4	0	0	3	0
pim	Pima	768	8	8	0	0	2	0
son	Sonar	208	60	60	0	0	2	0
tao	Tao	1,888	2	2	0	0	2	0
thy	Thyroid	215	5	5	0	0	3	0
veh	Vehicle	846	18	18	0	0	4	0
wbcd	Wisc. breast-cancer	699	9	0	9	0	2	2.3
wdbc	Wisc. diagnose breast-cancer	569	30	30	0	0	2	0
wne	Wine	178	13	13	0	0	3	0
wpbc	Wisc. prognostic breast-cancer	198	33	33	0	0	2	2
zoo	Zoo	101	17	0	1	16	7	0

methods to deal with continuous variables and missing values. IBk is an implementation of the nearest neighbor algorithm; it classifies a test instance with the majority class of its k nearest neighbors. Naive Bayes [33] is a probabilistic classifier that estimates the parameters of a Bayesian model. Part [19] is a learning architecture that combines the extraction of rules from partial decision trees and the separate-and-conquer rule learning technique to create a rule-based classifier without performing global optimization. SMO [39] is a widely-used implementation of *support vector machines* [46]. Zero-R is a very simple classifier that always predicts the majority class in the training dataset; this learner is included in the comparison to show a performance baseline. Table 2 summarizes the main characteristics of the learners.

All the non-fuzzy methods except for GAssist and UCS were run using WEKA [53]. For GAssist, we used the open source code provided in [3]. For UCS, we ran our own code. All the open source methods were configured with the parameters values recommended by default. Moreover, the models for SMO and IBk were selected as follows. For SMO, we ran the system with polynomial kernels of order 1, 3, and 10. Then, we ranked the results obtained with the three polynomial kernels and chose the model that maximized the average rank: SMO with polynomial kernels of order 3. Additionally, we also supply the results of SMO with Gaussian kernels. The same process was followed for IBk. We ran the experiments with k = $\{1, 3, 5\}$, and chose the configuration that maximized the average rank: k = 5. UCS was configured with the following parameters (see [6, 35] for notation details): $numIter$ = 100,000, N = 6,400, $acc_0 = 0.99$, $\nu = 10$, $\{\theta_{GA}, \theta_{del}, \theta_{sub}\} = 50$, $\chi = 0.8$, $\mu = 0.04$, $\delta = 0.1$, $P_{\#} = 0.6$.

Fuzzy AdaBoost and Fuzzy LogitBoost were run using KEEL [2]. Default parameters were used, except for the maximum population size which was set to 50. The results of SLAVE were supplied by the authors. They used exactly the same datasets and validation methodology as the other methods in the comparison. SLAVE used a steady-state GA with population size N = 100, and probabilities of crossover $\chi = 1.0$ and mutation $\mu = 0.4$. The GA run was stopped after 500 iterations without improvement. In all these fuzzy methods, we used five linguistic terms per variable, defined by triangular-shaped membership functions.

Comparison Metrics

The data models built by each learner were evaluated in terms of performance and interpretability. We measured the performance of the method with the test accuracy, i.e., the proportion of correct classifications on previously unseen examples. To obtain reliable estimates of the test accuracy, we used a ten-fold cross validation procedure [45].

The comparison of the interpretability of the models is more complicated since the methods included in the analysis use different knowledge representations. For this reason, we first identify two groups of learners. The first group

Table 2. Summary of the main characteristics of the learners included in the comparison: C4.5, IBk, Naive Bayes (NB), Part, SMO, Zero-R (0-R), UCS, GAssist (GAt), SLAVE (SLV), and the two boosting algorithms (Bst), Fuzzy AdaBoost and Fuzzy LogitBoost

	Paradigm	Knowledge Rep. and Inference Method
C4.5	Decision-tree induction	Decision-tree. *Inference*: class given by the corresponding leaf.
IBk	Instance-based learning	No general model. *Inference:* class determined by the majority class of the k nearest neighboors.
NB	Statistical Modeling	Probabilities of a Bayesian model. *Inference:* the output is the class with maximum probability.
Part	Rule induction based on decision-tree induction and a separate-and-conquer approach	Ordered list of rules. Continuous variables represented by float-coded attributes. *Inference:* the output is the class of the first matching rule in the ordered list.
SMO	Neural networks (support vector machines)	Weights of the support vector machines. *Inference:* The class is determined by the decision function represented by the SVM.
0-R	Majority-class predictor	No knowledge representation. *Inference:* Majority class in the training dataset.
UCS	Michigan-style GBML	Population of intervalar rules with a fitness or strength value. *Inference:* The output is the most voted class (weighted by fitness) among the matching rules.
GAt	Pittsburgh-style GBML	Ordered list of intervalar rules (intervals obtained via discretization). Use of a default rule. *Inference:* the output is the class of the first matching rule in the ordered list.
SLV	Iterative Rule GBML	Population of linguistic fuzzy rules. *Inference:* class determined by the rule with maximum matching.
Bst	Statistical Learning Theory and GBML	Population of linguistic fuzzy rules with a strength per class. *Inference:* The output is the most voted class (weighted by the strength) among the matching classifiers.

consists of those methods that build models that can be hardly interpreted (e.g., models represented with weights) and learners that do not create any data model, i.e., *lazy learners*. The second group comprises those learners that build interpretable models (e.g. trees and rule sets). We excluded the first group of learners from the comparison, and focused on the models created by the second group of learners. As this group consists of learners with

different types of knowledge representation, we provide some information of the sizes of the models to qualitatively compare them. For tree-based learners, we supply the number of leaves. For the rule-based systems, we provide the total number of rules evolved. Finally, we qualitatively compare these results considering the type of rule sets created and their size.

Statistical Analysis

We statistically analyzed the performance of each learner following the procedure pointed in [15]. As suggested by the author, we avoided to use any parametric statistical test since they require that the data satisfy several strong conditions. Instead, all the statistical analysis is based on non-parametric tests.

We first applied a multi-comparison statistical procedure to test whether all the learning algorithms performed the same on average. Specifically, we used the Friedman test [23, 24], the non-parametric equivalent to the analysis of variance test ANOVA [18]. If the Friedman test rejected that all the learners performed the same on average, post-hoc test were applied. Our first concern was to compare the performance of each GBML system with respect to the performance of the other learners. For this purpose, we applied the non-parametric Bonferroni–Dunn test [17]. The Bonferroni–Dunn test defines that one learner performs significantly differently to a control learner if the corresponding average rank differs by, at least, a critical difference CD computed as

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

where n_l is the number of learners, n_{ds} is the number of datasets, and q_α is the critical value based on the Studentized range statistic [43]. We illustrate the results of this test by showing the group of learners that perform equivalently to the control learner.

The Bonferroni–Dunn test is said to be conservative, specially as the number of learners increases or the number of datasets decreases, so that it may not detect significant differences although they actually exist. Nonetheless, we use this test in the first step of our analysis since it permits to detect groups of learners that truly perform differently from other learners. We later apply pairwise comparisons to detect further significant differences between learners that belong to the same group. We used the non-parametric Wilcoxon signed-ranks test [49] for pairwise comparisons, and provide the approximative p-values computed as indicated in [43].

4.2 Results

Comparison of the Performance

Table 3 details the test accuracies obtained by each learner on the 20 real-world problems. The average performance of Fuzzy AdaBoost for the problems

Table 3. Comparison of the test performance of all the machine learning techniques: C4.5, IB5, Naive Bayes (NB), Part, SMO with polynomial kernels of order 3 (SMO_3), SMO with Gaussian kernels (SMO_r), Zero-R (0-R), GAssist (GAt), SLAVE (SLV), Fuzzy AdaBoost (ABst), and Fuzzy LogitBoost (LBst). The two last rows show the average rank of each learning algorithm (Rnk), and its position in the rank (Pos)

	C4.5	IB5	NB	Part	SMO_3	SMO_r	0-R	UCS	GAt	SLV	ABst	LBst
ann	98.9	97.3	86.3	98.6	99.3	91.9	76.2	99.0	97.9	96.8	–	76.2
aut	80.9	64.0	58.8	74.4	78.1	45.5	32.6	77.4	68.6	70.7	–	32.6
bal	77.4	88.2	90.6	82.9	91.2	88.3	45.5	77.3	79.6	72.0	85.5	88.3
bpa	62.3	58.8	56.0	67.6	60.0	58.0	58.0	67.5	62.2	60.0	65.3	64.5
cmc	52.6	46.5	50.6	50.0	48.7	42.7	42.7	50.3	53.6	46.1	49.5	51.1
col	85.3	81.5	78.2	84.5	75.6	82.4	63.1	96.3	94.3	82.9	63.1	63.1
gls	66.1	64.7	48.9	66.6	66.1	35.7	35.7	70.0	65.1	57.6	62.5	68.2
h-c	78.5	83.2	82.8	74.2	78.6	82.5	54.5	79.7	80.1	77.9	60.4	62.1
h-s	79.3	80.7	83.3	80.0	78.9	82.6	55.6	74.6	77.7	76.3	57.6	59.3
irs	94.0	96.0	96.0	94.0	92.7	93.3	33.3	95.4	96.2	93.3	95.5	95.3
pim	74.2	73.3	75.8	74.9	76.7	65.1	65.1	74.6	73.8	72.3	70.7	71.8
son	71.1	84.0	69.7	74.4	85.5	69.3	53.4	76.5	75.8	73.1	46.6	53.4
tao	95.9	97.1	81.0	94.3	84.2	83.6	49.9	87.0	91.6	82.1	91.5	91.7
thy	94.9	94.8	97.2	94.3	88.9	69.8	69.8	95.1	92.5	91.3	97.4	97.1
veh	71.1	68.9	46.3	73.4	83.3	41.7	25.4	71.4	67.0	66.5	30.8	37.2
wbcd	95.0	97.1	96.1	95.7	96.4	96.1	65.5	96.3	95.6	96.4	94.9	94.1
wdbc	94.4	96.8	93.1	94.5	97.6	92.9	63.1	96.0	94.2	91.7	37.3	62.7
wne	93.9	96.7	97.2	93.3	97.7	39.9	39.9	96.1	93.2	94.3	85.6	85.0
wpbc	71.6	78.9	69.5	70.0	81.3	73.0	73.0	69.4	72.3	75.7	23.6	76.4
zoo	92.8	90.5	94.5	93.8	97.8	76.0	41.9	96.8	94.0	96.5	41.9	41.9
Rnk	***5.25***	***4.68***	***5.88***	***5.13***	***4.23***	***8.18***	***11.08***	***4.4***	***5.45***	***7.13***	***8.95***	***7.68***
Pos	*5*	*3*	*7*	*4*	*1*	*10*	*12*	*2*	*6*	*8*	*11*	*9*

ann and *aut* is not provided since the system was not able to extract competent fuzzy rules from the two domains, leaving nearly all the feature space uncovered. We confirmed with the authors that this behavior could be due to the high number of nominal attributes that these two problems have. The last two rows of the table supply the average rank and the position of each algorithm in the ranking. The ranks were calculated as follows. For each dataset, we ranked the learning algorithms according to their performance; the learner with the 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 experimental results show the competitiveness of UCS and GAssist. UCS presents the second best average rank; it is only outperformed by SMO with polynomial kernels of order 3. GAssist holds the sixth position of the ranking, degrading the average rank obtained by SMO with polynomial kernels, UCS, IB5, Part, and C4.5. Finally, SLAVE, Fuzzy LogitBoost, and Fuzzy AdaBoost are in the position 8, 9, and 11 of the ranking.

We analyzed if these differences in the average ranks of the learners were statistically significant by means of the multi-comparison test of Friedman. The statistical test rejected the hypothesis that all the methods performed the same on average with $p = 7.84 \cdot 10^{-12}$. To evaluate the differences among them, we applied different statistical tests. First, we compared the five GBML systems to all the other learners by means of a Bonferroni–Dunn test at a significance level of 0.1. Figure 2 graphically represents the groups of learners that perform equivalently to (a) UCS, (b) GAssist (GAt), (c) SLAVE (SLV), (d) Fuzzy LogitBoost (LBst), and (e) Fuzzy AdaBoost (ABst). We did not include Zero-R in the analysis, since its results are only provided as a baseline.

The statistical analysis confirms the robustness of UCS and GAssist. UCS belongs to the group of the best learners, significantly outperforming SLAVE, Fuzzy LogitBoost, SMO with Gaussian kernels, and Fuzzy AdaBoost. The test also indicates that GAssist does not significantly degrade the performance of SMO with polynomial kernels, the best ranked method. Moreover, GAssist significantly outperforms SMO with Gaussian kernels and Fuzzy AdaBoost.

As the Bonferroni–Dunn test is said to be quite conservative [15], so that it may not detect all the significant differences, we complemented the statistical study by comparing each pair of learners. Table 4 shows the approximate p-values of 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 $-$ are used to denote a non-significant improvement/degradation. Furthermore, Fig. 3 graphically illustrates the significant differences between methods. That is, each method is depicted in one vertex of the graph, and significant improvements (at $\alpha = 0.05$) of one learner with respect to another are plotted with a directed edge labeled with the corresponding p-value. To facilitate the visualization, the last three methods in

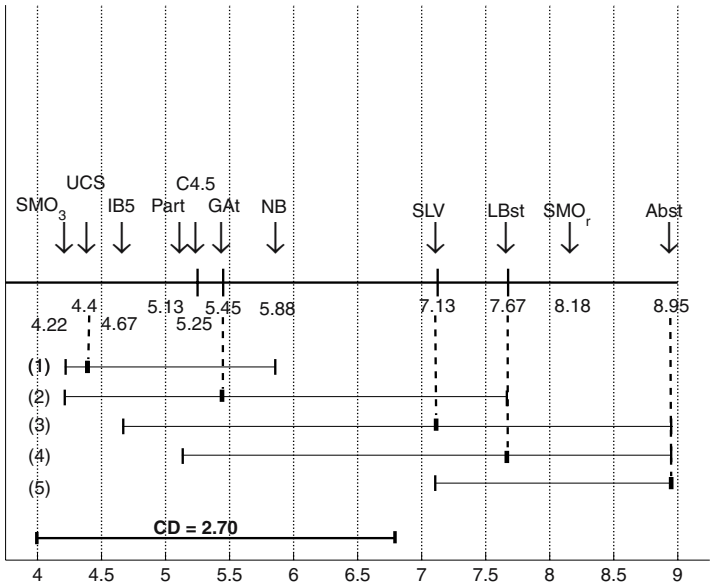

Fig. 2. Comparisons of one learner against the others with the Bonferroni–Dunn test at a significance level of 0.1. All the learners are compared to five different control groups: (1) UCS, (2) GAssist (GAt), (3) SLAVE (SLV), (4) Fuzzy LogitBoost (LBst), and (5) Fuzzy AdaBoost (ABst). The learners connected are those that perform equivalently to the control learner

Table 4. Pairwise comparisons of the learners by means of a Wilcoxon signed-ranks test. The above diagonal contains the approximate p-values. The below diagonal shows a symbol $\oplus$ / $\ominus$ if the method in the row significantly outperforms/degrades the method in the column at a significance level of .05 and +/=/− if there is no significant difference and performs better/equal/worse

	C4.5	IB5	NB	Part	SMO_{p3}	SMO_{rbf}	0-R	UCS	GAst	SLV	ABst	LBst
C4.5		.709	.263	.904	.421	.007	.000	.204	.941	.012	.001	.011
IB5	=		.053	.794	.412	.000	.000	.852	.455	.019	.002	.005
NB	−	−		.247	.067	.033	.000	.135	.108	.601	.003	.021
Part	+	−	+		.296	.006	.000	.232	.433	.014	.001	.006
SMO_3	+	+	+	+		.003	.000	.654	.296	.004	.003	.007
SMO_r	$\ominus$	$\ominus$	$\ominus$	$\ominus$	$\ominus$		.001	.006	.003	.027	.167	.576
0-R	$\ominus$	$\ominus$	$\ominus$	$\ominus$	$\ominus$	$\ominus$		.000	.000	.000	.249	.001
UCS	+	=	+	+	−	$\oplus$	$\oplus$		.218	.002	.002	.007
GAt	−	−	+	−	−	$\oplus$	$\oplus$	−		.017	.002	.017
SLV	$\ominus$	$\ominus$	+	$\ominus$	$\ominus$	$\oplus$	$\oplus$	$\ominus$	$\ominus$		.023	.057
ABst	$\ominus$	$\ominus$	$\ominus$	$\ominus$	$\ominus$	−	+	$\ominus$	$\ominus$	$\ominus$		.004
LBst	$\ominus$	$\ominus$	$\ominus$	$\ominus$	$\ominus$	−	$\oplus$	$\ominus$	$\ominus$	−	$\oplus$	

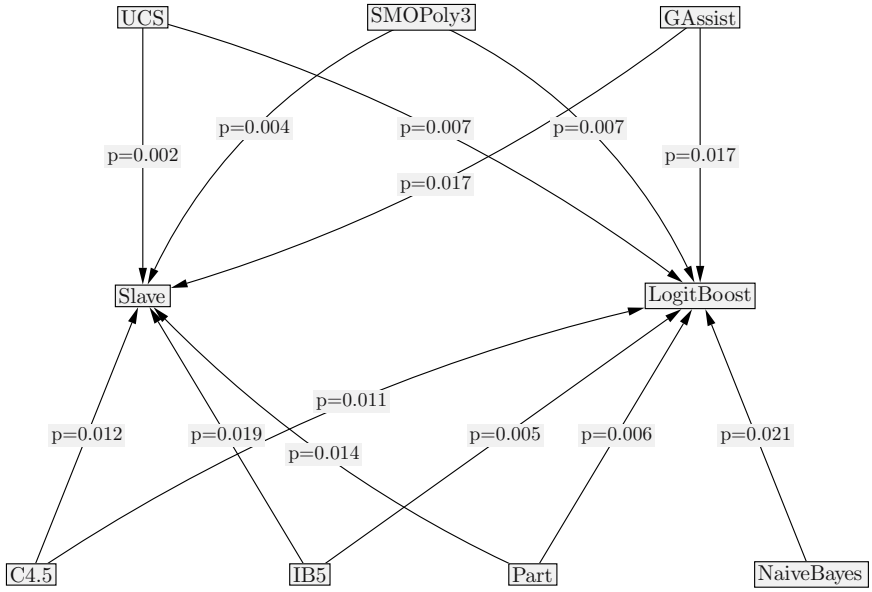

Fig. 3. Illustration of the significant differences of performance among methods. An edge $L_1 \overset{p_{value}}{\rightarrow} L_2$ indicates that the learner L_1 outperforms the learner L_2 with the corresponding p_{value}. To facilitate the visualization, SMO with Gaussian kernels, Fuzzy AdaBoost, and Zero-R are not included in the graph

the ranking are not included in the graph, i.e., SMO with Gaussian kernels, Fuzzy AdaBoost, and Zero-R. These three methods are outperformed by all the other methods.

The pairwise comparison confirms the conclusions derived from the Bonferroni–Dunn test, and finds further significant differences between pairs of learners. UCS, GAssist, SMO with polynomial kernels, Part, IB5, and C4.5 are the best methods in the comparison; any of them degrades the results obtained by other methods, and all them significantly outperform SLAVE, Fuzzy LogitBoost, SMO with Gaussian kernels, Fuzzy AdaBoost, and Zero-R. Thus, these results support that the two non-fuzzy GBML systems are, at least, as good as some of the most-used machine learning techniques.

The non-fuzzy GBML techniques present poorer results. Among them, the best method is SLAVE, with an average rank of 7.13. SLAVE is significantly outperformed by the six best methods of the analysis. Fuzzy LogitBoost is significantly outperformed by the same six methods and Naive Bayes. Fuzzy AdaBoost significantly degrades the results of all the other learners, except for SMO with Gaussian kernels, Fuzzy LogitBoost, and Zero-R. These results are not surprising. The linguistic fuzzy representation implies the discretization of the feature space, and the discretization points are fixed by the number of linguistic terms. In our experiments, we used only five linguistic

terms per variable. Thus, in this scenario, a single rule may not have the required granularity to define accurately the class boundaries of a given complex domain, limiting the maximum accuracy that can be achieved. However, the linguistic fuzzy representation allows for a better readability of the rules, as discussed in the next section.

Comparison of the Interpretability

Figure 4 plots partial examples of the models created by each learner (except for IBk, which does not create any model) for the two-dimensional *tao* problem. The picture highlights the differences between knowledge representations, which make difficult the comparison among them. In our study, we provide some characteristics of the models to qualitatively evaluate their readability.

We first distinguish between two types of learners (a) lazy learners or learners with knowledge representation based on weights, and (b) rule-based or tree-based learners. As before, we exclude Zero-R from the comparison. The first group consists of IBk, SMO, and Naive Bayes. IBk is a lazy classifier that does not create any global model from the training dataset; to predict the output of a new input example, IBk searches for the k nearest neighbors and returns the majority class among them. SMO represents the knowledge by $\binom{n}{2}$ support vector machines (where n is the number of classes of the classification problem), each one consisting of a set of real-valued weights (see Fig. 4a). Naive Bayes builds models formed by a set of parameters which estimate the independent probabilities of a Bayesian model. Consequently, the knowledge created by these three methods is really hard to interpret. Without a further comparison among them, we state that these three methods provide the poorest models in terms of interpretability.

The second group comprises the tree-based learner C4.5, and the rule-based systems Part, UCS, GAssist, SLAVE, Fuzzy AdaBoost, and Fuzzy LogitBoost. C4.5 evolves a tree in which each node represents a decision over one variable (see Fig. 4b). Part creates a set of rules which are defined by a conjunction of conditions over their variables, and are interpreted as an ordered activation list (see Fig. 4c). The knowledge representation of GAssist, UCS, SLAVE, Fuzzy AdaBoost, and Fuzzy LogitBoost is detailed in Sect. 3, and examples are shown in Figs. 4d–g. It is worth noting the differences between these systems. The rule sets evolved by GAssist are interpreted as an ordered activation list, similarly to Part; besides, GAssist uses a default rule. On the other hand, in UCS, SLAVE, Fuzzy AdaBoost, and Fuzzy LogitBoost, rules are not ordered; they represent independent classifiers. Consequently, all the matching rules participate in the classification of new input instances (except for SLAVE, the inference process of which only considers the rule with maximum matching degree).

To evaluate the size of the models, we used the following metrics. For the tree-based system, we counted the number of decision leaves. For the rule-based systems, we used the number of rules. Note that these measures are not

```
-    1.000    * <0.229 0.875> * X]
-    0.298    * <0.708 0.437 > * X]
                ...
```

(a) SMO

```
x <= -2.75
|  x <= -3.25: red (308.0)
|  x > -3.25
|  |  y <= 1.75: red (55.0)
|  |  y > 1.75
|  |  |  x <= -3: red (11.0/1.0)
|  |  |  x > -3
|  |  |  |     y < = 4.25: blue (6.0)
|  |  |  |     y > 4.25: red (4.0)
                ...
```

(b) C4.5

if x $\leq$ -3.25 **then** red (308)
else if x > 2.75 **then** blue (347/1)
else if y $\leq$ 0 **and** x $\geq$ -1 **then** red (192/1)
...

(c) Part

if x > 2.72 **and** (y is [0.92,4.61] or y > 5.07) **then** blue
else if (x is [-0.54, 0.54] or x > 2.72) **and** y is [-4.28, -2.57] **then** blue
...
otherwise red

(d) GAssist

if x is [-6.00, -0.81] **and** y is [-6.00, 0.40] **then** red **with** acc = 1.00
if x is [2.84, 6.00] **and** y is [-5.26, 4.91] **then** blue **with** acc = 1.00
if x is [-6.00, -0.87] **and** y is [-6.00, 0.74] **then** red **with** acc = 1.00
...

(e) UCS

if x is {XS or S or M} **and** y is {XS or S or M} **then** red
if x is XS **then** red
if x is {VL or M or H or VH} **then** blue
...

(f) SLAVE

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
...

(g) Boosting

Fig. 4. Examples of part of the models evolved by (**a**) SMO, (**b**) C4.5, (**c**) Part, (**d**) GAssist, (**e**) UCS, (**f**) SLAVE, and (**g**) Fuzzy AdaBoost and Fuzzy LogitBoost for the two-dimensional tao problem

Table 5. Average sizes of the models built by C4.5, Part, UCS, GAssist, SLAVE, Fuzzy AdaBoost (ABst), and Fuzzy LogitBoost (LBst)

	C4.5	Part	UCS	GAssist	SLAVE	ABst	LBst
ann	38	15	4,410	5	8	50	50
aut	44	21	4,064	7	17	50	50
bal	45	37	1,712	8	22	50	50
bpa	25	9	2,603	6	6	50	50
cmc	162	168	3,175	15	49	50	50
col	5	9	3,446	5	7	50	50
gls	24	15	3,013	5	15	50	50
h-c	29	21	2,893	6	6	50	50
h-s	17	18	3,499	5	7	50	50
irs	5	4	634	3	3	50	50
pim	19	7	3,225	7	13	50	50
son	14	8	5,999	5	9	50	50
tao	36	17	609	6	3	50	50
thy	8	4	1,283	4	5	50	50
veh	69	32	4,601	7	26	50	50
wbcd	12	10	1,799	3	5	50	50
wdbc	11	7	5,079	4	5	50	50
wne	5	5	3,413	3	4	50	50
wpbc	12	7	5,078	4	10	50	50
zoo	11	8	1,244	6	7	50	50

directly comparable due to the differences in the knowledge representations. However, we use these metrics to make a qualitative analysis.

Table 5 shows the model sizes of the rule-based and tree-based systems. These results show that:

- GAssist evolves rule sets that are significantly smaller than the rule sets created by all the other methods according to a Wilcoxon signed-rank test (at $\alpha = 0.05$). Thus, in terms of population size, GAssist is the best method in the comparison.
- SLAVE builds the second smallest rule sets, being only improved by GAssist. However, note that the types of rules evolved by SLAVE are much easier to read for two main reasons:
  1. GAssist uses an intervalar representation, where the intervals are obtained by applying a discretization technique over the input space. On the other hand, SLAVE uses a linguistic fuzzy representation. That is, variables are represented by linguistic terms; so, the rules can be easily read by human experts.
  2. GAssist uses an ordered activation list; that is, a rule is used to infer the class of a new input instance only if all the previous rules in the list do not match with this instance. Thus, the context of the rules (i.e., the conditions of the previous rules in the activation list) has to

be considered to read the whole rule set. Oppositely, in SLAVE, rules are independent classifiers. So, human experts can read each rule individually. This supposes one of the main advantages of the knowledge representation of SLAVE with respect to the representation of GAssist.

3. GAssist uses a default rule, which reduces considerably the rule set size, while SLAVE does not. The rule set sizes of SLAVE could be further reduced by including a default rule.

For these three reasons, we consider that the models created by SLAVE are more interpretable than those evolved by GAssist, even though the rule sets evolved by SLAVE are slightly bigger than those created by GAssist.

- Fuzzy AdaBoost and Fuzzy LogitBoost create rule sets of moderate size. In fact, the size of these rule sets is determined by a configuration parameter. In our experiments, we set the maximum population size to 50 since it maximized the average performance rank of the two learners. Smaller population sizes could be set for few specific problems without loss of accuracy. However, in our analysis, we are interested in robust methods that do not highly depend on the configuration parameters. For this reason, we used the same parameters in all runs, and did not search for the best configuration of each system for each particular dataset.

  This identifies a disadvantage of the two statistical methods with respect to the other learners. While UCS, GAssist, and SLAVE evolve a different number of rules depending on the intrinsic complexities of the domain, Fuzzy AdaBoost and Fuzzy LogitBoost need to know beforehand the number of rules to be created. Several techniques could be applied to overcome this drawback. For example, new approaches could be designed to remove the new rules that do not improve the accuracy of the compound classifier.
- UCS evolves the largest rule sets among all the methods in the comparison. Thus, even using a rule-based representation, the high number of rules may hinder the interpretability of the models evolved. Some reduction techniques have been proposed to remove non-useful rules from the final populations in XCS [16,25,52]. In further work, these reduction techniques will be adapted to UCS to try to improve the interpretability of the models.

The whole study performed through this section highlights that *genetic-based machine learning* is one of the best alternatives for facing challenging data mining and classification problems. The analysis also provides guidelines on which system should be used depending on the requirements of the problem. If the accuracy of the classification model is crucial, UCS appears to be the best approach to face a new problem. If the results also need to be readable, our recommendation is to use GAssist, since it offers highly interpretable models that are nearly as accurate as those created by UCS. If the readability prevails over the performance, SLAVE turns out to be the best choice as it creates a very low number of easily-readable independent linguistic fuzzy rules.

5 Summary and Conclusions

In this paper, we studied the suitability of *genetic-based machine learning* for pattern classification. For this purpose, we selected five GBML approaches that represent different styles and knowledge representations; two non-fuzzy GBML systems: UCS, GAssist; and three fuzzy GBML methods: SLAVE, Fuzzy AdaBoost, and Fuzzy LogitBoost. These learners were compared to six highly-used machine learning techniques on a set of 20 real-world problems. The results were analyzed by means of different statistical procedures.

The analysis showed the competence of GBML for classification tasks, and also pointed some recommendations on which GBML system use depending on the requirements of the user. UCS showed to be one of the best learners in terms of performance. It presented the second best performance, only being outperformed by the support vector machine SMO with polynomial kernels of order 3. GAssist resulted in a slightly inferior average performance, but evolved much more readable rule sets. SLAVE evolved the most interpretable models, but their performance was significantly inferior than the performance of the models created by UCS and GAssist.

Another observation drawn from the comparison is that there is not any method that outperforms another learner in all the problems. This indicates that the intrinsic complexities of the classification domains may have different effects depending on the learner. Our proposal as further work is to include measures to evaluate the complexity of the domains in the comparison, with the aim of pointing the complexities that affect the performance of the different GBML systems.

Acknowledgements

The authors thank the support of *Enginyeria i Arquitectura La Salle*, Ramon Llull University, as well as the support of *Ministerio de Educación y Ciencia* under project TIN2005-08386-C05-04, and *Generalitat de Catalunya* under Grants 2005FI-00252 and 2005SGR-00302.

References

1. J. Aguilar-Ruiz, J. Riquelme, and M. Toro. Evolutionary Learning of Hierarchical Decision Rules. *IEEE Transactions on Systems, Man, and Cybernetics Part B*, 33(2):324–331, 2003.
2. J. Alcalá-Fdez, M.J. del Jesus, J.M. Garrell, F. Herrera, C. Herbás, and L. Sánchez. Proyecto KEEL: Desarrollo de una herramienta para el análisis e implementación de algoritmos de extracción de conocimiento evolutivos. In J.S. Aguilar R. Giráldez, J.C. Riquelme, editor, *Tendencias de la Minería de Datos en España, Red Española de Minería de Datos y Aprendizage*, pages 413–424, 2004.

3. J. Bacardit. *GAssist Source Code: http://www.asap.cs.nott.ac.uk/jqb/PSP/GAssist-Java.tar.gz*.
4. J. Bacardit. *Pittsburgh Genetic-Based Machine Learning in the Data Mining Era: Representations, generalization and run-Time*. PhD thesis, Ramon Llull University, Barcelona, 2004.
5. T. Bäck. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, Oxford, 1996.
6. E. Bernadó-Mansilla and J.M. Garrell. Accuracy-Based Learning Classifier Systems: Models, Analysis and Applications to Classification Tasks. *Evolutionary Computation*, 11(3):209–238, 2003.
7. 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*, volume 2321 of *LNAI*, pages 115–132. Springer, Berlin Heidelberg New York, 2002.
8. C.L Blake and C.J. Merz. *UCI Repository of ML Databases: http://www.ics.uc.edu/mlearn/MLRepository.html*. University of California, 1998.
9. P. Bonelli and A. Parodi. An efficient classifier system and its experimental comparison with two representative learning methods on three medical domains. In *4th International Conference on Genetic Algorithms*, pages 288–295, 1991.
10. L. Castillo, A. González, and R. Pérez. Including a simplicity criterion in the selection of the best rule in a genetic fuzzy learning algorithm. *Fuzzy Sets and Systems*, 120:309–321, 2001.
11. O. Cordón, F. Herrera, F. Hoffmann, and L. Magdalena. *Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases*, volume 19 of *Advances in Fuzzy Systems–Aplications and Theory*. World Scientific, Singapore, 2001.
12. K.A. de Jong and W. Spears. Learning Concept Classification Rules Using Genetic Algorithms. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 651–656. Sydney, Australia, 1991.
13. K.A. de Jong, W.M. Spears, and D.F. Gordon. Using Genetic Algorithms for Concept Learning. *Genetic Algorithms for Machine Learning, A Special Issue of Machine Learning*, 13, 2–3, pages 161–188, 1993.
14. M.J. del Jesús, F. Hoffmann, L.J. Navascués, and L. Sánchez. Induction of fuzzy-rule-based classifiers with evolutionary boosting algorithms. *IEEE Transactions on Fuzzy Systems*, 12(3):296–308, 2004.
15. J. Demšar. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, 7:1–30, 2006.
16. P.W. Dixon, D.W. Corne, and M.J. Oates. *A Ruleset Reduction Algorithm for the XCSI Learning Classifier System*, volume 2661/2003 of *Lecture Notes in Computer Science*, pages 20–29. Springer, Berlin Heidelberg New York, 2004.
17. O.J. Dunn. Multiple Comparisons among Means. *Journal of the American Statistical Association*, 56:52–64, 1961.
18. R.A. Fisher. *Statistical Methods and Scientific Inference*, 2nd edn. Hafner Publishing Company, New York, 1959.
19. E. Frank and I.H. Witten. Generating accurate rule sets without global optimization. In *Proceedings of the 15th International Conference on Machine Learning*, pages 144–151. Morgan Kaufmann, San Francisco, 1998.
20. A. Freitas. *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Springer, Berlin Heidelberg New York, 2002.

21. Y. Freund and R.E. Schapire. Experiments with a New Boosting Algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.
22. J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 32(2):337–374, 2000.
23. M. Friedman. The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. *Journal of the American Statistical Association*, 32:675–701, 1937.
24. M. Friedman. A Comparison of Alternative Tests of Significance for the Problem of m Rankings. *Annals of Mathematical Statistics*, 11:86–92, 1940.
25. C. Fu and L. Davis. A modified classifier system compaction algorithm. In *GECCO'02: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 920–925. Morgan Kaufmann, San Francisco, 2002.
26. D.E. Goldberg. *Genetic Algorithms in Search, Optimization & Machine Learning*, 1st edn. Addison Wesley, Reading, 1989.
27. D.E. Goldberg. *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*, 1st edn. Kluwer, Boston, 2002.
28. A. Gónzalez and R. Pérez. Completeness and Consistency Conditions for Learning Fuzzy Rules. *Fuzzy Sets and Systems*, 96:37–51, 1998.
29. A. Gónzalez and R. Pérez. SLAVE: A Genetic Learning System Based on an Iterative Approach. *IEEE Transactions on Fuzzy Systems*, 7(2):176–191, 1999.
30. J.H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Michigan, 1975.
31. J.H Holland. Escaping Brittleness: The possibilities of General-Purpose Learning Algorithms Applied to Parallel Rule-Based Systems. In Michalski Mitchell and Carbonell, editors, *Machine Learning, an artificial intelligence approach*, volume II of *Lecture Notes in Artificial Intelligence*, pages 593–623. Morgan Kaufmann, San Francisco, 1986.
32. C.Z. Janikow. A Knowledge-Intensive Genetic Algorithm for Supervised Learning. *Machine Learning*, 13(2–3):189–228, 1993.
33. G.H. John and P. Langley. Estimating Continuous Distributions in Bayesian Classifiers. In *11th Conference on Uncertainty in Artificial Intelligence*, pages 338–345. Morgan Kaufmann, San Francisco, 1995.
34. Z. Liu, A. Liu, C. Wang, and Z. Niu. Evolving neural network using real coded genetic algorithm (GA) for multispectral image classification. *Future Generation Computer Systems*, 20(7):1119–1129, 2004.
35. A. Orriols-Puig and E. Bernadó-Mansilla. A Further Look at UCS Classifier System. In *GECCO'06: Genetic and Evolutionary Computation Conference Workshop Program*, ACM Press, Seattle, 08–12 July 2006.
36. J. Otero and L. Sánchez. Induction of descriptive fuzzy classifiers with the logitboost algorithm. *Soft Computing*, 10(9):825–835, 2006.
37. M. Pelikan. *Hierarchical Bayesian Optimization Algorithm: Toward a New Generation of Evolutionary Algorithms*, volume 170 of *Studies in Computational Intelligence*. Springer, Berlin Heidelberg New York, 2005.
38. M. Pelikan, K. Sastry, and E. Cantú-Paz. *Scalable Optimization via Probabilistic Modeling*, volume 33 of *Studies in Computational Intelligence*. Springer, Berlin Heidelberg New York, 2006.
39. J. Platt. Fast Training of Support Vector Machines using Sequential Minimal Opt. In *Advances in Kernel Methods - Support Vector Lear.* MIT Press, 1998.
40. J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, California, 1995.

41. J. Rissanen. Modeling by shortest data description. *Automatica*, vol. 14:465–471, 1978.
42. R.E. Schapire and Y. Singer. Improved Boosting Algorithms using Confidence-Rated Predictions. *Machine Learning*, 37(3):297–336, 1999.
43. D.J. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures*. Chapman & Hall, Boca Raton, 2000.
44. R.S. Sutton and A.G. Barto. *Reinforcement learning: An introduction*. MIT, Cambridge, 1998.
45. T.G. Dietterich. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation*, 10(7):1895–1924, 1998.
46. V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, Berlin Heidelberg New York, 1995.
47. G. Venturini. SIA: A Supervised Inductive Algorithm with Genetic Search for Learning Attributes Based Concepts. In P. B. Brazdil, editor, *Machine Learning: ECML-93 - Proceedings of the European Conference on Machine Learning*, pages 280–296. Springer, Berlin Heidelberg New York, 1993.
48. D. Wierstra, F.J. Gómez, and J. Schmidhuber. Modeling Systems with Internal State Using Evolino. In *GECCO'05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 1795–1802. ACM Press, New York, 2005.
49. F. Wilcoxon. Individual Comparisons by Ranking Methods. *Biometrics*, 1:80–83, 1945.
50. S.W. Wilson. Classifier Fitness Based on Accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.
51. S.W. Wilson. Generalization in the XCS Classifier System. In *3rd Annual Conference on Genetic Programming*, pages 665–674. Morgan Kaufmann, San Francisco, 1998.
52. S.W. Wilson. Compact Rulesets from XCSI. In *Advances in Learning Classifier Systems, 4th International Workshop*, volume 2321 of *Lecture Notes in Artificial Intelligence*, pages 197–210. Springer, Berlin Heidelberg New York, 2002.
53. I.H Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd edn. Morgan Kaufmann, San Francisco, 2005.