On Rule Learning Methods: A Comparative Analysis of Classic and Fuzzy Approaches

Marcos E. Cintra[1], Maria C. Monard[1], and Heloisa A. Camargo[2,⋆]

[1] University of Sao Paulo (USP) – Mathematics and Computer Science Institute
CEP 13566 - 590, Sao Carlos, SP
mecintra@gmail.com, mcmonard@icmc.usp.br
[2] Computer Science Department – Federal University of Sao Carlos
Rod. Washington Luis, Km 235, PO Box 676, 13565-905, Sao Carlos, SP, Brazil
heloisa@dc.ufscar.br

Abstract. Classification is an important task widely researched by the machine learning and fuzzy communities. In this paper, we present and compare methods from both communities, in order to support the selection of a suitable method, according to two conflicting objectives: accuracy × interpretability. Two groups of rule-based methods are analysed: decision tree-based and genetic-based approaches. For the tree-based approaches, C4.5, PART and FuzzyDT, a fuzzy version of the C4.5 algorithm, are used. For the genetic-based approaches, MPLCS, a method from the machine learning community to generate rule-based models, SLAVE and FCA-Based, both fuzzy-based, are analysed. Since accuracy and interpretability are usually conflicting objectives, in this paper, we briefly present these methods and then discuss the models generated by them. Comparisons take into account the error rates and syntactic complexity of the produced models. Ten benchmark datasets are used in the experiments with a 10 fold cross-validation strategy. Results show that FCA-Based and MPLCS are able to obtain good accuracy and interpretability.

Keywords: Fuzzy Systems, Decision Trees, Genetic Algorithms.

1 Introduction

Classification is an important task widely researched by the machine learning and fuzzy communities. Classic and fuzzy algorithms for supervised machine learning are concerned with the development of methods that extract patterns from data in order to make intelligent decisions based on these patterns. The interpretability is an important issue when classification methods are proposed. Interpretability of classification models, in spite of its subjectivity, can be defined as the quality of how easily the model, as a whole, can be understood and abstracted by its users. Thus, an approach that induces interpretable models must be concerned with the total number of rules and the amount of conditions

⋆ This work was supported by the Brazilian State Council FAPESP.

R.R. Yager et al. (Eds.): *Soft Computing: State of the Art Theory*, STUDFUZZ 291, pp. 89–104.
DOI: 10.1007/978-3-642-34922-5_7

of these rules, *i.e.*, the syntactic complexity of the model. In general, highly accurate models tend to have high syntactic complexity, whereas, models with low syntactic complexity tend to be less accurate.

In this sense, decision trees (DT) [1] are powerful as they produce models with low syntactic complexity which are quite intuitive and whose structures can be interpreted as rules. The induction process of DTs is usually fast and the induced models are competitive, accuracy wise, with the ones generated by other interpretable machine learning methods. Another desirable quality of DTs is their embedded feature selection process that allows it to use only the most relevant features in the model, which are selected according to certain measures, improving the generated model interpretability.

Some of the most well-known and relevant DT based algorithms are ID3, CART, and C4.5 [1, 2]. These algorithms generate a tree structure through recursively dividing the feature space until this decision space is completely partitioned into a set of non-overlapping subspaces. Specifically, C4.5 uses the information gain and entropy measures when deciding on the importance of the features [2]. In order to optimize their estimated error rates, DTs usually use a pruning process. Pruning also simplifies the whole models, which consequently become more interpretable. PART [3] is an example of a DT-based approach for rule generation. This method repeatedly generates various DTs extracting the best rule of each DT at a time to construct the rule set of a classifier.

Fuzzy rule based classification systems are based on the fuzzy set and fuzzy logic theories proposed by Loft A. Zadeh. Two advantageous characteristics of fuzzy systems regarding interpretability are: i) the system uses semantically meaningful fuzzy sets to define attributes; ii) fuzzy rules are built by linguistic variables and linguistic terms, such as "temperature is high" or "speed is low", adding interpretability to the induced model.

The knowledge base and inference mechanism are the two basic components of a fuzzy classification system. The knowledge base is formed by the Fuzzy Data Base (FDB) and the Fuzzy Rule Base (FRB). The fuzzy data base contains the definitions of the features (also named attributes or variables) in terms of fuzzy sets, while the fuzzy rule base contains a set of rules defining the given problem. The inference mechanism derives the conclusions (or outputs) of the system based on the knowledge base and on the inputs to the system.

In the literature, it is possible to find several fuzzy approaches for the induction of fuzzy classifiers, amongst them, fuzzy rule-based systems [4, 5], genetic fuzzy rule-based systems [6–8], fuzzy DTs [9, 10], and evolutionary (other than genetic) fuzzy rule-based systems [11–13]. Regarding the genetic rule-based fuzzy systems, their advantages include: i) Genetic Algorithms (GAs) perform a global search and do not get stuck in local maxima; ii) it is possible to address the interpretability $\times$ accuracy problem during the search process by means of multi-objective fitness functions; iii) it is possible to adjust rules and fuzzy sets during the genetic process in order to improve the model performance and interpretability [14, 15].

Although the genetic generation of fuzzy systems might be one of the most researched topic in the fuzzy community, GAs usually have a high computational cost due to their global search and, for some approaches, also, due to the process required to form the genetic search space. A well-known genetic fuzzy approach is SLAVE – Structural Learning Algorithm on Vague Environment [16]. SLAVE uses the iterative approach to learn fuzzy rules, performing an embedded feature selection process as well as a rule selection post process.

An alternative to the high cost of GAs is the DT-based approach. For this purpose, we have recently proposed a fuzzy version of the classic C4.5 DT in [17]. Our approach is quite similar to the classic one and its more relevant characteristics are described in Section 2.

The aim of this paper is to experimentally compare different proposals based on DTs and GAs, from both, the machine learning and fuzzy communities. Since there is a large number of genetic fuzzy approaches proposed, we selected two of them, the well known approach named SLAVE, and another one proposed by us, FCA-Based. On the other hand, due to the special characteristics of low computational cost, competitive and highly interpretable induced models, we also include two classic DT-based approaches and a fuzzy DT in our experiments. Comparisons were performed taking into account the accuracy and syntactic complexity of the generated models. The goal of this research is to provide substantial information on these approaches, indicating their most relevant qualities and drawbacks.

The remainder of this paper is organized as follows. Section 2 describes the FCA-Based, SLAVE and MPLCS methods, which use the genetic paradigm. Section 3 describes and compares the classic and the fuzzy C4.5 DT approaches, as well as PART. Section 4 presents the experiments and results, followed by the conclusions and future work in Section 5.

2 Classification Methods Based on the Genetic Paradigm

GAs [18] are a part of the evolutionary algorithms, which are techniques inspired on the biological evolution. GAs have been applied in several areas. They usually require a randomly generated initial population of hypotheses. For rule-based classification, the initial population is usually formed by rules or rule sets. The hypothesis of GAs is that the fittest members of a population have better chances of producing offspring. This way, by generating several populations, it is possible to evolve solutions and reach satisfactory results. This population, whose members are called chromosomes or individuals, encodes candidate solutions to a given problem. The first population gives rise to the following population by means of genetic operators, such as selection, mutation, crossover, and elitism, among others. At each generation, the hypotheses in the current population are evaluated relative to a given fitness measure, with the fittest hypotheses selected probabilistically as seeds to produce the next generation, and so on. Usually, GAs stop by reaching a maximum number of generations or when a satisfactory fitness level is reached.

Chromosomes are usually represented by an array of elements. These arrays can contain indexes to a preselected list of solutions (called search space). By representing hypotheses using arrays of elements with fixed length, the crossover operator is simple to apply due to the alignment of the chromosomes forming the population. Variable length representations are also used, but they require higher computational effort for the use of genetic operators. The fitness function evaluates the quality of the represented solution and it is always directly connected to the type of problem to be tackled. For classification problems using rule bases, the usual fitness measures adopted are related to the accuracy and interpretability of the generated models.

Next, we present the SLAVE, FCA-Based, and MPLCS fuzzy classification systems, which are the GA-based learning methods to generate classification rules used in this work.

2.1 Fuzzy Classification Systems

The classification task can be roughly described as: given a set of objects $E = \{e_1, e_2, ..., e_n\}$, also named *examples, cases, or instances*, which are described by m features, assign a class c_i from a set of classes $C = \{c_1, c_2, ..., c_j\}$ to an object e_p, described by its feature values $e_p = (a_{p_1}, a_{p_2}, ..., a_{p_m})$.

Fuzzy classification systems are rule based fuzzy systems that granulate the domains of their features by means of fuzzy sets and partitions. The linguistic variables in the antecedent part of the rules represent features, and the consequent part represents a class. A typical fuzzy classification rule can be expressed by

R_k : **IF** X_1 is A_{1l_1} **AND** X_2 is A_{2l_2} **AND** ... **AND** X_m is A_{ml_m}
THEN $Class = c_i$

where R_k is the rule identifier, $X_1, ..., X_m$ are the features of the set of examples considered in the problem (represented by linguistic variables), $A_{1l_1}, ..., A_{ml_m}$ are the linguistic values used to represent the feature values, and $c_i \in C$ is the class. Notice that not all identifiers participate in a general classification rule. The inference mechanism compares the input example to each rule in the fuzzy rule base aiming at determining the class it belongs to.

The classic and general fuzzy reasoning methods are widely used. Given a set of fuzzy rules, *i.e.*, a FRB, and an input instance, the classic fuzzy reasoning method classifies this input instance using the class of the rule with maximum compatibility to the input instance, while the general fuzzy reasoning method calculates the sum of compatibility degrees for each class and uses the class with highest sum to classify the input instance.

2.2 SLAVE

SLAVE [16] is a genetic learning algorithm that uses the iterative approach to generate a FRB. In the iterative approach, chromosomes usually represent individual rules, and a single rule is selected at each iteration of the GA. The set

of selected rules form the rule base of the model. SLAVE includes an embedded feature selection process. The preselection of attributes minimizes the problems caused by large search spaces, such as excessive execution time, while improving the interpretability of the generated models.

The main idea of SLAVE is to reduce the original problem of obtaining a complete set of rules to a simpler problem which consists in obtaining only one rule at a time. In this approach, each chromosome of the population represents a single rule, but only the best individual in each iteration is considered, the remaining chromosomes being discarded. In fact, in the iterative model, one execution of the GA provides a partial solution (a rule) to the learning problem.

Regarding the feature selection process adopted by SLAVE, it dynamically explores the set of possible variables in order to find the most useful rule and the most relevant variables for this rule. Thus, this feature selection process is implemented for each single rule, not for the whole set of rules. The basic schema of this process consists of modifying the rule representation in the search mechanism of SLAVE in order to allow the learning algorithm to search not only for the best rule, but also the best set of variables for each rule. SLAVE produces rules with different weights, which are used by the inference mechanism to improve the classification performance. SLAVE usually produces reasonably small rule sets.

2.3 FCA-Based

The FCA-Based method [8] forms the GA search space by using the theory of Formal Concept Analysis (FCA) [19]. FCA is a mathematical technique for extracting concepts and structures from data. It was introduced in the 1980s and is becoming increasingly popular due to its nice visual representation of data and relations found in data. The basic data structure in FCA is the formal context, which is a representation of the relations between objects and attributes. A formal context is usually represented in a table form where the columns represent the attributes and the rows represent the objects (objects are usually called instances or examples in classification). The most important difference between an attribute $\times$ value table and a formal context is that FCA only works with binary attributes. In order to handle continuous and multi-valued attributes, they must be transformed into binary attributes using a scaling process. The table representing the formal context contains 1 (true) in cell (i, j) if object i has attribute j, and 0 (false) otherwise. By extracting classification rules from data using the FCA theory to form the GA search space, the FCA-Based method is able to avoid the creation of a large number of useless rules, a task that has a high computational cost. After the rule extraction process, the FCA-Based method uses a GA to generate the fuzzy rule base.

The FCA-Based method uses an integer chromosome codification. The size of the chromosome is equal to the maximum number of rules considered acceptable for the final FRBs and it is initialized with the number of rules found in the rule base produced by the Wang & Mendel method [20] using the same FDB. This heuristic allows the definition of chromosomes with a reasonable number of

rules.This approach requires a considerable extra computational cost compared to our approach. The integer codification uses the index of a rule in the search space generated by FCA in each of its genes. To allow the generation of rule bases with less rules than the maximum size of the chromosome, a ***-1*** value is used to indicate that a gene represents an inactive rule.

For the fitness calculation, and aiming at reducing the number of rules in the final FRB, the FCA-Based method uses the Correct Classification Rate (CCR) and the number of rules (NR) in the rule base represented by each chromosome during the search process. This evaluation process uses a self-adaptive algorithm that keeps and updates referential values of the ideal CCR and NR. After each generation, an update occurs if a better CCR is obtained with a number of rules equal or smaller than the best current NR. In the sequence, the NR is used in a penalization mechanism that decreases the fitness value of a chromosome when its NR is larger than the current reference NR.

In order to improve the interpretability of the final rule bases generated by the GA, FCA-Based has a simple post selection process that checks the ability of each individual rule to improve the classification power of the rule set. This process aims at removing as many rules as possible while keeping (or improving) the accuracy of the whole FRB.

2.4 MPLCS

MPLCS [21] stands for Memetic[1] Pittsburgh Learning Classifier System (MPLCS). The MPLCS method has many variants according to the adopted local search mechanism.

The version used in our experiments uses the local search, based on the rule set-wise operator. This local search has three main stages: i) an evaluation of the candidate rules; ii) the selection of the rules that will form the offspring rule set; iii) the generation of the final individual. In the first stage, all rules are evaluated with all the examples of the training set, producing a map of correct and incorrect classifications for each rule. The next stage uses this map to evaluate how much each candidate rule can contribute to improve the accuracy of the offspring rule set without re-evaluating the rule set.

2.5 Comparing the Models Generated by the SLAVE, FCA-Based, and MPLCS

One issue with the models generated by SLAVE is that they contain rules with sets of fuzzy label disjunctions in their antecedents. For example, one of the models generated for the Iris dataset, with three rules, is presented next.

1. If **X2** is {L0}, class is **Iris-setosa** (W 0.977)

[1] Memetics is a theory of mental content based on an analogy with the Darwinian evolution. Memes are similar to genes in GA, but represent ideas, believes, patterns of behaviour, which can reproduce.

2. If **X0** is {L0 L1} and **X2** is {L0 L1} and **X3** is {L0 L1}, class is **Iris-versicolor** (W 0.402)
3. If **X0** is {L1 L2} and **X2** is {L1 L2} and **X3** is {L0 L2}, class is **Iris-virginica** (W 0.719)

Although the model has only 3 rules, with a total of 7 conditions, 6 of these conditions contain disjunctions of fuzzy labels, impacting on the model interpretability. It is also possible to find the association of linguistic values that are not defined by neighbouring fuzzy sets, which makes each rule quite difficult do understand.

The models generated by MPLCS, similarly to SLAVE, contain conjunctions of disjunctions, and, for continuous attributes, the splitting points can be quite unnatural and difficult to be interpreted. As an example, the rule set for the Iris dataset, with 4 rules, is shown next.

1. If **sepalLength** is > 6.243 and **petalLength** is > 5.085, class is **Iris-virginica**
2. If **sepalLength** is < 6.340 and > 7.020 and **petalWidth** is > 1.627, class is **Iris-virginica**
3. If **petalLength** is < 1.983, class is **Iris-setosa**
4. Default rule: class is **Iris-versicolor**

The cutting points defined by the algorithm can be quite similar and close to each other. For instance, the splitting points for **sepalLength** in rules 1 and 2 discard values from a very close interval from 6.243 to 6.340, which makes the understanding of the whole model difficult.

The FCA-Based models present only conjunctions of conditions in the antecedent of their rules, and, due to the fact that it is based on the fuzzy logic, the discretization of continuous attributes is done using highly interpretable linguist valued fuzzy sets. The rule set for the iris dataset is presented next.

1. If **sepalLenght** is **medium**, and **petalLenght** is **medium** and **petalWidth** is **medium**, **class** is **Iris-virginica**
2. If **petalLenght** is **large** and **petalWidth** is **medium**, **class** is **Iris-versicolor**
3. if **petalLength** is **small**, and **petalWidht** is **medium**, and **sepalWidth** is **small**, **class** is **Iris-setosa**
4. if **petalLength** is **medium**, and **petalWidht** is **small**, and **sepalWidth** is **large**, **class** is **Iris-virginica**

For experts and persons who are familiar with the domain, the fuzzy linguistic values **small**, **medium** and **large** are directly interpreted. Nevertheless, for those who are unfamiliar with the domain, it is necessary to check the FDB for the information regarding the number, type and distribution of the fuzzy sets defining each attribute, in order to interpret the rules and rule set. This process, although quite straightforward, requires extra effort. To reduce the time to understand the FDB, the information can be presented in graphs.

Next, we discuss DT based classification methods.

3 Classification Methods Based on Decision Trees

DTs are widely used in machine learning due to its simplicity of generation and powerful representation of knowledge. Fuzzy DTs have also been proposed in the literature. The classic C4.5 DT algorithm, PART (a DT-based approach for rule generation), and FuzzyDT, our fuzzy version of the C4.5 algorithm, are presented next, as well as a comparison of their generated models.

3.1 C4.5

DT algorithms generate a tree structure through recursively partitioning the feature space until the whole decision space is completely divided into a set of non-overlapping class subspaces (leaf nodes). They also perform an embedded selection of features during its partitioning process, so only relevant features are used in the tree, improving the time used to classify new examples as well as the interpretability of the model. C4.5 is one of the most well-know DT algorithms [2]. C4.5 uses the information gain and entropy measures when deciding on the importance of the features.

In order to avoid overfitting, a stopping criterion can be used to prevent some subsets of training examples from being subdivided. The pruning of a part of the DT structure helps preventing overfitting. Regarding the pruning process, C4.5 employs post-pruning, *i.e.*, the pruning takes place after the tree is completely induced assessing the error rates of the tree and its components directly on the set of training examples [1]. This assessment is related to the confidence level that the error obtained with the pruned tree, in relation to the error for the original tree, will represent the real error.

The default confidence level used by C4.5 is 25%. It is important to notice that the smaller the confidence limits, the higher the chances of pruning, while the higher the confidence limits, the smaller the chances of pruning. Thus, if we set the confidence limit to 100%, what we are saying is that we believe that the predicted error, obtained with the examples at hand, is equal to the real error and no pruning will be performed. This idea conflicts with the natural intution one might have that a 25% confidence limit will produce less pruning than an 80% confidence limit. This way, one should not associate the default 25% confidence limits of C4.5 with a 25% pruning of the tree.

3.2 PART

PART, as its name indicates, is an algorithm based on partial DTs [3]. A partial DT is an ordinary DT that contains branches to undefined subtrees. PART is a rule-induction procedure that adopts the separate-and-conquer strategy. In essence, it builds a rule, removes the instances it covers, and continues creating rules recursively for the remaining instances until none is left. In order to generate this rule, PART generates a DT and prunes all but one leaf (specifically the leaf with the largest coverage) and makes the branch of this leaf into a rule, discarding the rest of the tree. Its authors explain that using a pruned tree to obtain a rule,

instead of building it incrementally by adding conjunctions of conditions one at a time, avoids the over-pruning problem of the basic separate-and-conquer rule learner.

In fact, using the separate-and-conquer methodology in conjunction with DTs adds flexibility and speed to the process. Since it is wasteful to build a full DT just to obtain a single rule, PART significantly accelerates the process described without sacrificing the above advantages by building a "partial" DT instead of a fully explored one. An integration of the construction and pruning operations is used in order to find a "stable" subtree that can be simplified no further. This way, once this subtree has been found, the tree induction ceases and a single rule is selected.

3.3 FuzzyDT

FuzzyDT [17] is a fuzzy implementation of the classic C4.5 algorithm. It uses the same measures of C4.5 (entropy and information gain) to decide on the importance of the features. The main difference between the classic and the fuzzy C4.5 is the fact that the fuzzy version discretizes continuous attributes using fuzzy sets before the induction of the tree.This way, the process can be seen as inducing a tree using only discrete features, since the continuous features are defined in terms of fuzzy sets and the training set is fuzzified before the DT induction.

3.4 Comparing the Models Generated by the C4.5, PART, and FuzzyDT

The model produced by C4.5, as by most of the DT algorithms, form a set of disjunct rules in which only one rule is fired to classify a new example. For FuzzyDT, on the other hand, the tree can be seen as a set of rules that are fired simultaneously. Since they are fuzzy rules, the degree of compatibility of each rule with a new example is calculated and used by the inference mechanism to classify this new example. This way, the inference of fuzzy DTs requires higher computational effort than the classic DTs. In spite of this additional cost, the compatibility information of the rule with the example guides the inference, which considers all attributes of a rule (branch), while classic DTs simply tests one attribute at a time, even if the input values are close to the test values.

PART generates a set of ordered rules. The inference process is quite straightforward: the first rule is checked, if it does not cover the example, the next rule is checked, until the example is classified or the last rule is reached.

Next we compare some important features and definitions of FuzzyDT, C4.5 and PART.

Evaluation of features – For the partitioning process, the three methods use the same measures, *i.e.*, entropy and information gain, to select the features to be used in the test nodes of their branches or rules;

Induction process – FuzzyDT and C4.5 repeatedly subdivide the feature space using the most informative features until a leaf node is reached or no

features or examples remain. PART uses a similar approach to generate partial trees, but for each generated tree, only the branch of the tree that correctly classifies the largest number of examples is used. This process is repeated as many times as necessary.

Handling continuous features – PART and C4.5 split the domain according to the examples at hand by minimizing entropy and maximizing information gain. The drawback of this process is the discretization of continuous attributes, which might create unnatural divisions that reflect on a lower interpretability of the rules and rule set. Another issue with PART and C4.5 is that the number of divisions used to split continuous attributes, even if known a priori, cannot be informed to the algorithm. In fact, the splitting of continuous attributes is done dynamically by the algorithms, and might be distant from the patterns of the application domain or even from the representation used by an expert. FuzzyDT, on the other hand, can use the partitions (and thus, number of fuzzy sets) defined by an expert. Furthermore, even if this information is not available, fuzzy partitions can be automatically defined and are easily interpretable. The equalized universe method [22], which evenly splits the domain into a defined number of fuzzy sets, is a simple approach that prevents the creation of unnatural splitting points.

Reuse of features – for PART and C4.5, the same continuous feature can be used more than once in one single rule (for example, if the feature is temperature, a rule might present tests such as "temperature ≤ 95", "temperature ≤ 74", "temperature ≤ 10", and so on). This repetition of the same feature and subdivision of the domain degrades the interpretation of the rule. On the other hand, asFuzzyDT fuzzyfies ("discretizes") the attributes using fuzzy sets, a feature can be used only once in one rule, favoring the interpretability of the generated rules.

Inference – The C4.5 algorithm checks the root test and then the following triggered branch of the tree, to classify a new example. The process is intuitive and clear. Similarly, PART checks the ordered rule set in sequence. However, for continuous features, whenever the input values are located in the decision frontiers, misclassifications might occur due to the fact that the whole inference is done based on a single attribute at a time. For FuzzyDT, as stated before, the membership degree of the input example is calculated for each fuzzy set defining each attribute. These membership degrees are then used to calculate the confidence degree for each rule. Since all branches might be fired simultaneously, this confidence degree is used by the classification process, taking into consideration all the attributes included in each rule, instead of the approach used by the classic DTs of checking a single attribute at a time. This way, FuzzyDT gives more credibility to the final classification. Nevertheless, although many branches of the tree might not be fired by an example, if the DT is large, the FuzzyDT inference process will require a considerable computational effort when compared to PART and C4.5. The calculation can be reduced by defining a minimum threshold of membership degree to continue testing rules or not.

Next, we present the experiments and results.

4 Experiments

Our experiments were carried out using 10 datasets from the UCI Machine Learning Repository [23] and 10-fold cross-validation strategy. The KEEL framework [24] was used for the SLAVE, C4.5, PART, and MPLCS algorithms, all executed with default parameters, except for the number of fuzzy sets for SLAVE, which was set to 3. For FCA-Based and FuzzyDT, we used our own implementations.

As previously stated, the motivation to compare such different approaches for the automatic generation of classifiers is to provide information and insight when selecting a classification method for a particular problem. For this purpose, we considered the performance of the models, in terms of error rates, and their interpretability, in terms of their syntactic complexity, which takes into consideration the average number of rules generated and the average number of conditions of these rules.

Table 1 summarizes the dataset characteristics giving the total number of examples (Examples); number of features (Features), including the number of continuous and discrete features in brackets; number of classes (Classes), and the majority error (ME), which is the error of the most naive algorithm, which always predicts the majority class of the dataset.

Table 1. General characteristics of the datasets

Dataset	Examples	Features	Classes	ME
Credit	653	15(6,9)	2	45.33
Cylinder	277	32(19,13)	2	35.74
Diabetes	769	8(8,0)	2	34.90
Glass	220	9(9,0)	7	65.46
Heart	270	13(13,0)	2	44.44
Ionosphere	351	34(34,0)	2	35.90
Iris	150	4(4, 0)	3	66.67
Segment	210	19(19,0)	7	85.72
Vehicle	846	18(18,0)	4	74.23
Wine	178	13(13,0)	3	59.74

Notice that all fuzzy methods used the same (FCA-Based and FuzzyDT) or similar (SLAVE) fuzzy partitions, *i.e.*, the same number of fuzzy sets and their distribution, as well as the type of membership function.

Table 2 presents the mean error rates and standard deviation for the tested methods. The majority error (ME) is presented in the second column. The lasts lines present the average rank and the final rank for each approach. For the average rank and the final rank for each method, when computing the error rank for each dataset, if two or more error measures are equal, the fractional strategy for assigning rankings was used, *i.e.*, they receive the same ranking numbers, which is the mean of what they would have under ordinary rankings. The results for the DT-based approaches are presented in the first columns, and thus, the results for the GA-based approaches in the last columns. The smallest error rates are dark-gray shaded.

Table 2. Error rates and standard deviation for decision tree based approaches

Approaches		DT-based approaches						GA-based approaches					
		FuzzyDT		C4.5		PART		SLAVE		FCA-Based		MPLCS	
Datasets	ME	Error	SD	Error	SD	Error	SD	Error	SD	Error	SD	Error	SD
Credit	45.33	15.78	6.68	12.09	4.39	37.35	13.54	37.38	3.86	9.82	3.90	11.02	4.89
Cylinder	35.74	34.20	0.09	27.69	10.95	31.32	10.68	35.74	0.96	25.77	5.15	25.95	10.85
Diabetes	34.90	26.10	0.05	22.89	8.07	31.47	10.49	25.66	4.26	23.09	2.30	23.67	8.41
Glass	65.46	48.26	8.12	27.03	11.29	48.98	17.50	38.25	10.19	39.56	5.21	29.39	12.17
Heart	44.44	21.85	5.60	20.37	8.32	38.15	13.86	20.74	10.10	19.72	7.44	17.41	8.61
Iono	35.90	13.53	9.95	11.11	4.51	20.52	13.02	12.27	4.81	11.76	4.97	9.98	4.66
Iris	66.67	8.00	2.67	4.00	5.33	60.00	20.00	4.67	4.27	4.68	6.33	2.67	5.33
Segment	85.71	20.48	5.65	0.48	0.45	10.70	4.88	13.43	0.43	24.75	7.39	0.69	0.55
Vehicle	74.23	35.85	4.05	24.33	9.09	58.36	19.96	39.59	4.82	44.15	4.73	27.04	10.64
Wine	59.74	12.86	11.43	6.67	6.94	36.31	14.51	9.54	7.05	4.27	4.99	5.00	3.89
Final Rank		4.9		2.3		5.6		3.3		3.0		1.9	
Avg. Rank		5		2		6		4		3		1	

Considering only the DT-based approaches, C4.5 obtained the smallest error rates for all datasets. FuzzyDT presented smaller error rates than PART for 8 datasets. It should be observed that in most cases learning was very poor for PART, as well as for FuzzyDT using Cylinder, *i.e.*, the error rate for these models is similar to the most naive learning algorithm that always predicts the most frequent class in the dataset.

Considering only the GA-based approaches, MPLCS had the smallest error rates for 6 datasets while FCA-Based for 4 datasets. Moreover, the error rate of the model generated by SLAVE for Cylinder is the ME, thus, there was no learning.

Comparing all methods, *i.e.*, DT-based and GA-based, C4.5 obtained the smallest error rates for 4 datasets, while FCA-Based and MPLCS for three datasets each.

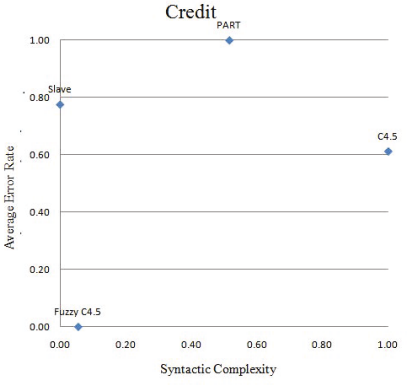
To test whether there was a statistically significant difference among the six algorithms we used the Friedman test [25] with the null-hypothesis that the performance of all algorithms, assessed in terms of the error rates, was comparable. The Friedman test found there is no statistically significant difference among the tested algorithms with a 95% confidence level.

As discussed previously, some methods tend to present good error rates, but low interpretability, or vice-versa. This way, to analyse the interpretability of the generated models, Table 3 presents the average number of rules and the Syntactic Complexity (SC) of the models generated by the six analysed algorithms, as well as the average rank and final rank of their SC. In this work, the SC is defined as the total number of conditions in each rule set. The dark-gray shaded cells highlight the smallest syntactic complexity values obtained in both approaches. Notice that although the rules produced by SLAVE and MPLCS present conjunctions of disjunctions, Table 3 does not consider the number of disjunctions in the rules of these models.

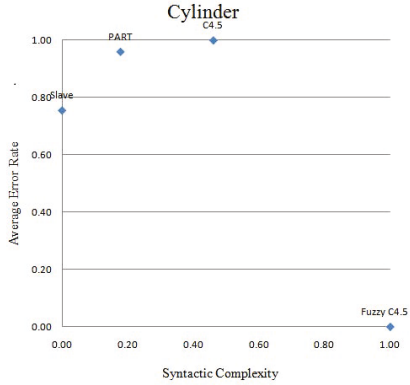
As one can observe, PART produced the models with the smallest syntactic complexity for 9 of the datasets and SLAVE for the remaining one. Observe that dataset Cylinder is the remaining one, which had the ME as error rate (Table 3).

Table 3. Average number of rules and syntactic complexity values

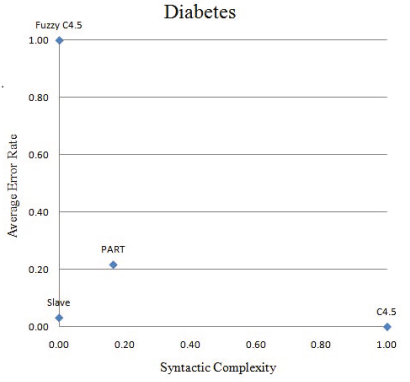
Approaches	DT-based approaches						GA-based approaches					
	FuzzyDT		C4.5		PART		SLAVE		FCA-Based		MPLCS	
Datasets	Rules	SC	Rules	SC	Rules	SC	Rules	SC	Rules	SC	Rules	SC
Credit	21.0	64.5	19.6	95.4	3.0	12.4	5.1	18.6	10.5	32.8	6.9	33.9
Cylinder	31.8	83.7	42.8	248.5	3.6	17.6	1.0	1.0	15.8	70.1	11.8	118.4
Diabetes	12.6	34.4	23.6	150.2	1.3	3.7	4.2	19.6	9.1	33.1	8.3	33.6
Glass	26.6	95.0	24.1	137.8	2.5	10.1	11.9	51.1	6.8	39.4	7.6	21.9
Heart	17.4	49.0	18.5	86.1	1.4	4.0	8.3	43.9	14.1	58.5	7.0	30.7
Iono	20.2	54.4	13.9	72.4	2.4	9.8	15.1	73.0	19.9	77.5	4.6	19.5
Iris	8.2	13.4	4.6	12.1	1.0	2.0	3.2	10.4	4.5	12.8	4.0	7.6
Segment	22.6	72.2	10.0	38.0	1.3	3.2	3.5	15.9	11.5	49.2	4.2	9.3
Vehicle	65.6	296.9	66.3	503.0	2.7	12.0	21.7	151.9	30.2	172.9	19.2	72.9
Wine	13.8	35.0	5.1	12.5	1.9	5.7	4.5	30.2	4.9	15.3	4.2	6.6
Final Rank	4.9		5.1		1.1		3.1		4.1		2.7	
Avg. Rank	5		6		1		3		4		2	



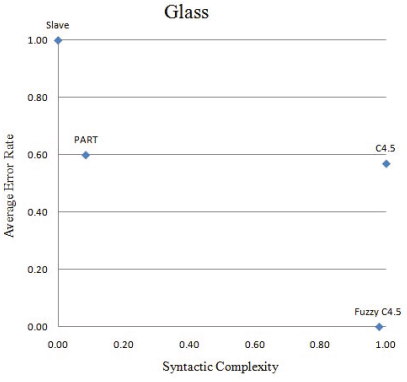
(a) Credit (error[9.8, 37.3], SC[11.2, 95.4])



(b) Cylinder (error[25.8, 34.2], SC[1.0, 248.5])



(c) Diabetes (error[22.9, 31.5], SC[3.7, 150.2])



(d) Glass (error[27.0, 49.0], SC[10.1, 137.8])

Fig. 1. Error × SC for Credit, Cylinder, Diabetes, and Glass

Thus, the model generated is simply one rule assigning the majority class to any new instance. However, PART was ranked last in accuracy (Table 2). On the other hand, although C4.5 is ranked last regarding the SC, it is ranked second for accuracy. Furthermore, MPLCS is ranked second regarding the SC, and it is ranked first for accuracy.

In order to consider the performance of the methods both in terms of error rates and syntactic complexity, we used the normalized values of the error rate and SC to produce some graphs and visually analyse the results. To illustrate, Figure 4 presents the results for Credit, Cylinder, Diabetes, and glass, the first four datasets. Because the values are normalized, notice that the origins of the graphs do not represent null error and null syntactic complexity. Instead, the origins are defined by the smallest error rate and SC of the results of the tested methods for each dataset. Similarly, point (1,1) represents the maximum error and syntactic complexity obtained on the dataset. By using the normalized values, instead of the real ones, it is easier to choose the most appropriate algorithms for a specific dataset by focusing on the ones that are plotted closest to the origin of the graphs. The idea is to discard the methods whose values are plotted farthest from the origin and just compare and analyse those closest to the origin in order to obtain the best compromise between error rate and syntactic complexity.

For datasets Credit and Diabetes, the FCA-Based algorithm presents the smallest error rate and low SC. The second best would be MPLCS. However, it is important to notice that the rules produced by MPLCS contain the disadvantage of being formed by conjunctions of disjunctions, while FCA-Based produce quite clear and interpretable rules. For the Cylinder dataset, FCA-Based should be chosen, and MPLCS for the Glass dataset. C4.5, although having low error rates, had the worst SC for these datasets. PART and SLAVE had both good SC, but poor error rates.

Next, we present the final conclusions.

5 Conclusions

Classification is an important task in the machine learning and fuzzy communities. Many classification approaches have been proposed by both communities, some of them sharing similar cores. For instance, both communities have decision tree-based methods, genetic-based methods, methods based on artificial neural networks, among others. Aiming at comparing similar methods from both communities that produce interpretable models, two groups of rule-based methods are analysed in this work: decision tree-based and genetic-based approaches.

The decision tree-based group include C4.5, PART and FuzzyDT. The genetic-based group includes MPLCS, a method from the machine learning community to generate rule-based models, as well as SLAVE and FuzzyDT, both fuzzy-based. These methods were analysed according to their accuracy and syntactic complexity on ten benchmark datasets using a ten fold cross-validation strategy.

Results show that FCA-Based and MPLCS were able to obtain good accuracy and interpretability, while the other methods had good accuracy and poor syntactic complexity, or poor accuracy and good syntactic complexity. One important issue when comparing the models produced by FCA-Based and MPLCS is the fact that MPLCS, as well as SLAVE, produce rules with conjunctions of conditions which might contain sets of disjunctions. This characteristic makes MPLCS and SLAVE much less complex with respect to the SC than the ones produced by C4.5, PART, and FCA-Based, although the disjunctions impact on the readability of the rules, instead of improving it.

As future work, we intend to include other methods from both communities in the experiments and consider other important issues in our comparisons, such as the time taken to generate the models, and their ability to classify examples from datasets whose classes have different cost for misclassification, such as in medical domains. We also intend to use a larger set of datasets.

References

1. Quinlan, J.R.: C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning), 1st edn. Morgan Kaufmann (January 1993)
2. Quinlan, J.R.: Bagging, Boosting and C4.5. In: Proc. of the 13th Conf. on Artificial Intelligence, pp. 725–730 (1996)
3. Frank, E., Witten, I.H.: Generating accurate rule sets without global optimization. In: ICML 1998: Proceedings of the 15th Int. Conf. on Machine Learning, pp. 144–151. Morgan Kaufmann (1998)
4. Ishibuchi, H., Yamamoto, T.: Rule weight specification in fuzzy rule-based classification systems. IEEE Transactions on Fuzzy Systems 13, 428–435 (2005)
5. Nakashima, T., Schaefer, G., Yokota, Y., Ishibuchi, H.: A weighted fuzzy classifier and its application to image processing tasks. Fuzzy Sets and Systems 158, 284–294 (2007)
6. Mansoori, E., Zolghadri, M., Katebi, S.: Sgerd: A steady-state genetic algorithm for extracting fuzzy classification rules from data. IEEE Transactions on Fuzzy Systems 16(4), 1061–1071 (2008)
7. Cintra, M.E., Camargo, H.A.: Fuzzy rules generation using genetic algorithms with self-adaptive selection. In: IEEE International Conference on Information Reuse and Integration - IRI, pp. 261–266 (2007)
8. Cintra, M.E., Monard, M.C., Camargo, H.A., Martin, T.P.: A hybrid approach for the automatic generation of fuzzy systems using fuzzy formal concepts. In: 2012 IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2012 (accepted for publication, 2012)
9. Olaru, C., Wehenkel, L.: A complete fuzzy decision tree technique. Fuzzy Sets and Systems 138(2), 221–254 (2003)
10. Cintra, M.E., Monard, M.C., Camargo, H.A.: An evaluation of rule-based classification models induced by a fuzzy method and two classic learning algorithms. In: The Brazilian Symposium on Artificial Neural Network (SBRN), vol. 1, pp. 188–193 (2010)
11. Ahmadizar, F., Soltanpanah, H.: Reliability optimization of a series system with multiple-choice and budget constraints using an efficient ant colony approach. Expert Systems with Applications 38(4), 3640–3646 (2011)

12. Marinaki, M., Marinakis, Y., Stavroulakis, G.E.: Fuzzy control optimized by a multi-objective particle swarm optimization algorithm for vibration suppression of smart structures. Struct. and Multidisciplinary Optimization 43(1), 29–42 (2011)
13. Prakash, A., Deshmukh, S.: A multi-criteria customer allocation problem in supply chain environment: An artificial immune system with fuzzy logic controller based approach. Expert Systems with Applications 38(4), 3199–3208 (2011)
14. Angelov, P.P.: An evolutionary approach to fuzzy rule-based model synthesis using indices for rules. Fuzzy Sets and Systems 137, 325–338 (2003)
15. Chiou, Y., Lan, L.W.: Genetic fuzzy logic controller: an iterative evolution algorithm with new encoding method. Fuzzy Sets and Systems 152, 617–635 (2005)
16. Gonzalez, A., Perez, R.: Selection of relevant features in a fuzzy genetic learning algorithm. IEEE Transactions on Systems and Man and and Cybernetics and Part B: Cybernetics 31(3), 417–425 (2001)
17. Cintra, M.E., Camargo, H.A.: Feature Subset Selection for Fuzzy Classification Methods. In: Hüllermeier, E., Kruse, R., Hoffmann, F. (eds.) IPMU 2010. CCIS, vol. 80, pp. 318–327. Springer, Heidelberg (2010)
18. Mitchell, T.M.: Machine Learning. McGraw-Hill (1997)
19. Wille, R.: Restructuring lattice theory: An approach based on hierarchies of concepts. In: Rivals, I. (ed.) Ordered Sets, vol. 23, pp. 445–470 (1982)
20. Wang, X.Z., Wang, Y.D., Xu, X.F., Ling, W.D., Yeung, D.S.: A new approach to fuzzy rule generation: fuzzy extension matrix. Fuzzy Sets and Systems 123, 291–306 (2001)
21. Bacardit, J., Krasnogor, N.: Performance and efficiency of memetic pittsburgh learning classifier systems. Evolutionary Computation 17(3), 307–342 (2009)
22. Chen, M.S., Wang, S.W.: Fuzzy clustering analysis for optimizing membership functions. Fuzzy Sets and Systems 103, 239–254 (1999)
23. Frank, A., Asuncion, A.: UCI machine learning repository (2010)
24. Bull, L., Bernadó-Mansilla, E., Holmes, J.: Learning Classifier Systems in Data Mining. Springer (2008)
25. Demsar, J.: Statistical comparison of classifiers over multiple data sets. Journal of Machine Learning Research 7(1), 1–30 (2006)