International Journal of Approximate Reasoning 51 (2010) 729–747

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

journal homepage: www.elsevier.com/locate/ijar

A fuzzy random forest

Piero Bonissone [a], José M. Cadenas [b,*], M. Carmen Garrido [b], R. Andrés Díaz-Valladares [c]

[a] GE Global Research, One Research Circle, Niskayuna, NY 12309, USA
[b] Dept. Ingeniería de la Información y las Comunicaciones, Universidad de Murcia, Spain
[c] Dept. Ciencias Computacionales, Universidad de Montemorelos, Mexico

ARTICLE INFO

Article history:
Received 6 September 2009
Received in revised form 10 February 2010
Accepted 10 February 2010
Available online 16 February 2010

Keywords:
Random forest
Fuzzy decision tree
Combination methods
Fuzzy sets

ABSTRACT

When individual classifiers are combined appropriately, a statistically significant increase in classification accuracy is usually obtained. Multiple classifier systems are the result of combining several individual classifiers. Following Breiman's methodology, in this paper a multiple classifier system based on a "forest" of fuzzy decision trees, i.e., a fuzzy random forest, is proposed. This approach combines the robustness of multiple classifier systems, the power of the randomness to increase the diversity of the trees, and the flexibility of fuzzy logic and fuzzy sets for imperfect data management. Various combination methods to obtain the final decision of the multiple classifier system are proposed and compared. Some of them are weighted combination methods which make a weighting of the decisions of the different elements of the multiple classifier system (leaves or trees). A comparative study with several datasets is made to show the efficiency of the proposed multiple classifier system and the various combination methods. The proposed multiple classifier system exhibits a good accuracy classification, comparable to that of the best classifiers when tested with conventional data sets. However, unlike other classifiers, the proposed classifier provides a similar accuracy when tested with imperfect datasets (with missing and fuzzy values) and with datasets with noise.

1. Introduction

Classification has always been a challenging problem [1,14]. The explosion of information that is available to companies and individuals further compounds this problem. There have been many techniques and algorithms addressing the classification issue. In the last few years we have also seen an increase of multiple classifier systems based approaches, which have been shown to deliver better results than individual classifiers [27]. However, imperfect information inevitably appears in realistic domains and situations. Instrument errors or corruption from noise during experiments may give rise to information with incomplete data when measuring a specific attribute. In other cases, the extraction of exact information may be excessively costly or unviable. Moreover, it may on occasion be useful to use additional information from an expert, which is usually given through fuzzy concepts of the type: small, more or less, near to, etc. In most real-world problems, data have a certain degree of imprecision. Sometimes, this imprecision is small enough for it to be safely ignored. On other occasions, the imprecision of the data can be modeled by a probability distribution. Lastly, there is a third kind of problem where the imprecision is significant, and a probability distribution is not a natural model. Thus, there are certain practical problems where the data are inherently fuzzy [9,28,30,31].

* Corresponding author. Tel.: +34 868 884847; fax: +34 868 884151.
E-mail addresses: bonissone@ge.com (P. Bonissone), jcadenas@um.es (J.M. Cadenas), carmengarrido@um.es (M. Carmen Garrido), rdiaz@um.edu.mx (R. Andrés Díaz-Valladares).

0888-613X/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.ijar.2010.02.003

Therefore, it becomes necessary to incorporate the handling of information with attributes which may, in turn, present missing and imprecise values in both the learning and classification phases of the classification techniques. In addition, it is desirable that such techniques be as robust as possible to noise in the data.

In this paper, we will focus on how to start from a multiple classifier system with performance comparable to or better than the best classifiers and extend it to handle imperfect information (missing values and fuzzy values) and make it robust to noise in nominal attributes and to outliers in numerical attributes [6,10]. To build the multiple classifier system, we follow the random forest methodology [8], and for the processing of imperfect data, we construct the random forest using a fuzzy decision tree as base classifier. Therefore, we try to use the robustness of both, a tree ensemble and a fuzzy decision tree, the power of the randomness to increase the diversity of the trees in the forest, and the flexibility of fuzzy logic and fuzzy sets for imperfect data management.

The majority vote is the standard combination method for random forest ensembles. If the classifiers in the ensemble are not of identical accuracy, then it is reasonable to attempt to endow the more "competent" classifiers with more power in making the final decision when using weighted majority vote. In this work, we propose and compare various weighted combination methods to obtain the final decision of the proposed multiple classifier system.

In Section 2, we review the major elements that constitute a multiple classifier system, providing a brief description of how the outputs of each classifier are combined to produce the final decision and we comment on some aspects of the incorporation of fuzzy logic in classification techniques. In Section 3, we explain the learning and classification phases of the proposed multiple classifier system, which we call a fuzzy random forest. In Section 4, we define combination methods for the fuzzy random forest. In Section 5, we show different computational results that illustrate the behaviour of the fuzzy random forest. Finally, we present our conclusions in Section 6.

2. Multiple classifier systems and fuzzy logic

When individual classifiers are combined appropriately, there is usually a better performance in terms of classification accuracy and/or speed to find a better solution [1]. Multiple classifier systems are the result of combining several individual classifiers. Multiple classifier systems differ in their characteristics by the type and number of base classifiers; the attributes of the dataset used by each classifier; the combination of the decisions of each classifier in the final decision of the ensemble; and the size and the nature of the training dataset for the classifiers.

2.1. Decision tree-based ensembles

In recent years several ensemble techniques have been proposed using different base classifiers. However, this work focuses on ensembles using decision trees as base classifier. Therefore, we will present them in chronological order to show the evolution of this concept in the literature.

Bagging [7] is allegedly one of the oldest techniques for creating an ensemble of classifiers. In bagging, diversity is obtained by constructing each classifier with a different set of examples, which are obtained from the original training dataset by re-sampling with replacement. Bagging then combines the decisions of the classifiers using uniform-weighted voting.

The boosting algorithm [15,32] creates the ensemble by adding one classifier at a time. The classifier that joins the ensemble at step k is trained on a dataset sampled selectively from the original dataset. The sampling distribution starts from uniform and progresses towards increasing likelihood of misclassified examples in the new dataset. Thus, the distribution is modified at each step, increasing the likelihood of the examples misclassified by the classifier at step $k-1$ being in the training dataset at step k.

Ho's random subspaces technique [19] selects random subsets of the available attributes to be used in training the individual classifiers in the ensemble.

Dietterich [13] introduced an approach called randomization. In this approach, at each node of each tree of the ensemble, the 20 best attributes to split the node are determined and one of them is randomly selected for use at that node.

Finally, Breiman [8] presented random forest ensembles, where bagging is used in tandem with random attribute selection. At each node of each tree of the forest, a subset of the available attributes is randomly selected and the best split available within those attributes is selected for that node. The number of attributes randomly chosen at each node is a parameter of this approach.

In a recent paper [3], Banfield et al. compared these decision tree ensemble creation techniques. They proposed an evaluation approach using the average ranking of the algorithms on each dataset.

2.2. Combination methods

In [24,25] some of the views in the literature about combination of classifiers are described. In this paper, we follow one of these views of grouping the combination methods in multiple classifier systems into trainable and non-trainable combiners.

Non-trainable combiners are those that do not need training after the classifiers in the ensemble have been trained individually. Trainable combiners may need training during or after the training of individuals ones. In the literature the trainable combiners also are called data-dependent combiners and are divided into implicity dependent and explicitly dependent. The implicity data-dependent group contains trainable combiners where the parameters of the combiner do not depend on the target example. In other words, the parameters are trained before the system is used for classifying new examples. The explicit data-dependent combiners use parameters that are functions of the target example.

2.3. Fuzzy logic in classification techniques

Although decision tree techniques have proved to be interpretable, efficient and capable of dealing with large datasets, they are highly unstable when small disturbances are introduced in training datasets. For this reason, fuzzy logic has been incorporated in decision tree construction techniques.

Leveraging its intrinsic elasticity, fuzzy logic offers a solution to overcome this instability. In [21–23,26,29] we find approaches in which fuzzy sets and their underlying approximate reasoning capabilities have been successfully combined with decision trees. This integration has preserved the advantages of both components: uncertainty management with the comprehensibility of linguistic variables, and popularity and easy application of decision trees. The resulting trees show an increased robustness to noise, an extended applicability to uncertain or vague contexts, and support for the comprehensibility of the tree structure, which remains the principal representation of the resulting knowledge.

Thus, we propose a random forest with a fuzzy decision tree as base classifier. Among the various techniques of ensemble based on decision trees, we have chosen random forest because, like boosting, it generates the best results [3]. In addition, as concluded in [8], random forest is more noise resistant (when a fraction of values of the class attribute in the training dataset are randomly altered) than boosting based ensembles. Therefore, we take advantage of the improvement in results that provide multiple classifier systems compared to the individual classifiers, and increase the noise resistance of random forests based ensembles to use fuzzy decision trees instead of crisp decision trees as base classifier. In addition, the use of fuzzy decision trees adds to random forest some of the advantages that we have commented on earlier for this type of technique: uncertainty management with the comprehensibility of linguistic variables, an increased noise resistance, and an extended applicability to uncertain or vague contexts.

3. Fuzzy random forest: an ensemble based on fuzzy decision trees

Following Breiman's methodology, we propose a multiple classifier system that is a random forest of fuzzy decision trees. We will refer to it as a fuzzy random forest ensemble and it will be denoted as FRF ensemble. In this section, we describe the learning phase required to construct the multiple classifier system and its classification phase.

In the random forest proposed by Breiman, [8], each tree is constructed to the maximum size and without pruning. During the construction process of each tree, every time that it needs to split a node (i.e. select a test at the node), only a random subset of the total set of available attributes is considered and a new random selection is performed for each split. The size of this subset is the only significant design parameter in the random forest. As a result, some attributes (including the best) might not be considered for each split, but an attribute excluded in one split might be used by other splits in the same tree. Random forests have two stochastic elements [8]: (1) bagging is used for the selection of the datasets used as input for each tree; and (2) the set of attributes considered as candidates for each node split. These randomizations increase the diversity of the trees and significantly improve their overall predictive accuracy when their outputs are combined. When a random forest is constructed, about 1/3 of the examples are excluded from the training dataset of each tree in the forest. These examples are called "out of bag" (OOB) [8]; each tree will have a different set of OOB examples. The OOB examples are not used to build the tree and constitute an independent test sample for the tree [8].

3.1. Fuzzy random forest learning

We propose Algorithm 1 to generate a random forest whose trees are fuzzy decision trees, so defining, a basic algorithm to generate the FRF ensemble.

Algorithm 1. FRF ensemble Learning

```
FRFlearning(in: E, Fuzzy Partition; out: Fuzzy Random Forest)
begin
  1. Take a random sample of $|E|$ examples with replacement from the dataset $E$
  2. Apply Algorithm 2 to the subset of examples obtained in the previous step to construct a fuzzy tree, using the fuzzy partition
  3. Repeat steps 1 and 2 until all fuzzy trees are built to constitute the FRF ensemble
end
```

Each tree in the FRF ensemble will be a fuzzy tree generated along the guidelines in [22], modifying it so as to adapt it to the functioning scheme of the FRF ensemble. Algorithm 2 shows the resulting algorithm.

Algorithm 2. Fuzzy Decision Tree Learning

```
**FuzzyDecisionTree**(in: E, Fuzzy Partition; out: Fuzzy Tree)
begin
  1. Start with the examples in E with values $\chi_{Fuzzy\_Tree,root}(e) = 1$
  2. Let M be the set of attributes where all numeric attributes are partitioned according to the Fuzzy Partition
  3. Choose an attribute to do the split at the node N
    3.1. Make a random selection of attributes from the set of attributes M
    3.2. Compute the information gain for each selected attribute using the values $\chi_{Fuzzy\_Tree,N}(e)$ of each e in node N
    3.3. Choose the attribute such that information gain is maximal
  4. Divide N in children nodes according to possible outputs of the attribute selected in the previous step and remove it from the
   set M. Let $E_n$ be the dataset of each child node
  5. Repeat steps 3 and 4 with each $(E_n, M)$ until the stopping criteria is satisfied
end
```

Algorithm 2 has been designed so that the trees can be constructed without considering all the attributes to split the nodes. We select a random subset of the total set of attributes available at each node and then choose the best one to make the split. So, some attributes (including the best one) might not be considered for each split, but an attribute excluded in one split might be used by other splits in the same tree. Algorithm 2 is an algorithm to construct trees based on ID3, where the numeric attributes have been discretized through a fuzzy partition. This study uses the fuzzy partitioning algorithm for numerical attributes proposed in [11]. The domain of each numeric attribute is represented by trapezoidal fuzzy sets, $A_1, \ldots, A_f$ so each internal node of the tree, whose division is based on a numerical attribute, generates a child node for each fuzzy set of the partition. The fuzzy partition of each attribute guarantees completeness (no point in the domain is outside of the fuzzy partition), and is a strong fuzzy partition (satisfying that $\forall x \in E, \sum_{i=1}^{f} \mu_{A_i}(x) = 1$, where $A_1, \ldots, A_f$ are the fuzzy sets of partition given by its membership functions μ_{A_i}).

Moreover, Algorithm 2 uses a function, which we will call $\chi_{t,N}(e)$, which indicates the degree with which the example e satisfies the conditions that lead to node N of tree t. This function is defined as follows:

- Each example e used in the training of the tree t has been assigned an initial value 1 ($\chi_{t,root}(e) = 1$) indicating that this example was initially found only in the root node of tree t.
- According to the membership degree of the example e to different fuzzy sets of partition of a split based on a numerical attribute, the example e may belong to one or two children nodes, i.e., the example will descend to a child node associated with membership degree greater than 0 ($\mu_{fuzzy\_set\_partition}(e) > 0$). In this case $\chi_{t,childnode}(e) = \chi_{t,node}(e) \times \mu_{fuzzy\_set\_partition}(e)$.
- When the example e has a missing value on the attribute used as split in a node, the example e descends to each child node with a modified value $\chi_{t,childnode}(e) = \chi_{t,node}(e) \times \frac{1}{number\_outputs_{split}}$.

The stopping criterion in Algorithm 2 is triggered by the first encountered condition among the following ones: (1) a node is pure, i.e., node that contains examples of only one class, (2) the set of available attributes is empty, (3) the minimum number of examples allowed in a node has been reached. When we build the FRF ensemble with the above algorithm, we obtain the OOB set for each fuzzy tree.

With Algorithms 1 and 2, we integrate the concept of fuzzy tree within the design philosophy of Breiman's random forest.

3.2. Fuzzy random forest classification

In this section, we will describe how the classification is carried out using the FRF ensemble. First, we introduce the notation that we will use. Then, we define two general strategies to obtain the decision of the FRF ensemble for a target example. Concrete instances of these strategies are defined in the next section, where we present different combination methods for the FRF ensemble.

3.2.1. Notations

We describe the notation necessary to define the strategies and combination methods used by the FRF ensemble.

- T is the number of trees in the FRF ensemble. We will use the index t to refer to a particular tree.
- N_t is the number of leaf nodes reached by an example, in the tree t. A characteristic inherent in fuzzy trees is that in classification an example can reach two or more leaves due to the overlapping of the fuzzy sets that constitute the partition of a numerical attribute. We will use the index n to refer to a particular leaf reached in a tree.
- I is the number of classes. We will use the index i to refer to a particular class.
- e is an example which will be used either as an example of training or as a test.
- $\chi_{t,n}(e)$ is the degree of satisfaction with which example e reaches the leaf n from t tree, as we indicated in Section 3.1.

- *Support* for the class i is obtained in each leaf as $\frac{E_i}{E_n}$ where E_i is the sum of the degrees of satisfaction of the examples with class i in leaf n and E_n is the sum of the degrees of satisfaction of all examples in that leaf.
- *L_FRF* is a matrix with size $(T \times MAX_{N_t})$ with $MAX_{N_t} = max\{N_1, N_2, \ldots, N_T\}$, where each element of the matrix is a vector of size I containing the support for every class provided by every activated leaf n on each tree t. Some elements of this matrix do not contain information since not all the trees of the forest have MAX_{N_t} reached leaves. Therefore, the matrix *L_FRF* contains all the information generated by the FRF ensemble when it is used to classify an example e and from which it makes its decision or class with certain methods of combination.

$L\_FRF_{t,n,i}$ refers to an element of the matrix that indicates the support given to the class i by the leaf n of tree t. An example of the matrix assuming that $I = 2,\ T = 3,\ N_1 = 1,\ N_2 = 2$ and $N_3 = 1$ may be:

(0.7,0.3)	—
(0.2,0.8)	(0.3,0.7)
(0.1,0.9)	—

The information in this matrix is directly provided by each fuzzy tree of the FRF ensemble. From this matrix, we can get new information, through certain transformation of the same, which we will use later in some combination methods. The transformations that we will use are:

- *Transformation 1, denoted as TRANS*1: This transformation, $L\_FRF = TRANS1(L\_FRF)$, provides information where each reached leaf assigns a simple vote to the majority class. For example, if we apply this transformation to the previous matrix get the following matrix:

$$\begin{array}{|c|c|}\hline (0.7,0.3) & — \\ \hline (0.2,0.8) & (0.3,0.7) \\ \hline (0.1,0.9) & — \\ \hline \end{array} \Rightarrow \text{TRANS1} \Rightarrow \begin{array}{|c|c|}\hline (1,0) & — \\ \hline (0,1) & (0,1) \\ \hline (0,1) & — \\ \hline \end{array}$$

- *Transformation 2, denoted as TRANS*2: This transformation, $L\_FRF = TRANS2(L\_FRF)$, provides information where each reached leaf votes with weight $\chi_{t,n}(e)$ for the majority class. For example, if we apply this transformation to the previous matrix and $\chi_{1,1}(e) = 1,\ \chi_{2,1}(e) = 0.6,\ \chi_{2,2}(e) = 0.4$ and $\chi_{3,1}(e) = 1$ we get the following matrix:

$$\begin{array}{|c|c|}\hline (0.7,0.3) & — \\ \hline (0.2,0.8) & (0.3,0.7) \\ \hline (0.1,0.9) & — \\ \hline \end{array} \Rightarrow \text{TRANS2} \Rightarrow \begin{array}{|c|c|}\hline (1,0) & — \\ \hline (0,0.6) & (0,0.4) \\ \hline (0,1) & — \\ \hline \end{array}$$

- *Transformation 3, denoted as TRANS*3: This transformation, $L\_FRF = TRANS3(L\_FRF)$, provides information where each reached leaf provides support for each class weighted by the degree of satisfaction with which the example e reaches the leaf. For example, if we apply this transformation to the previous matrix and $\chi_{1,1}(e) = 1,\ \chi_{2,1}(e) = 0.6,\ \chi_{2,2}(e) = 0.4$ and $\chi_{3,1}(e) = 1$ we get the following matrix:

$$\begin{array}{|c|c|}\hline (0.7,0.3) & — \\ \hline (0.2,0.8) & (0.3,0.7) \\ \hline (0.1,0.9) & — \\ \hline \end{array} \Rightarrow \text{TRANS3} \Rightarrow \begin{array}{|c|c|}\hline (0.7,0.3) & — \\ \hline (0.12,0.48) & (0.12,0.28) \\ \hline (0.1,0.9) & — \\ \hline \end{array}$$

- *T_FRF* is a matrix with size $(T \times I)$ that contains the confidence assigned by each tree, t, to each class i. The matrix elements are obtained from the support for each class in the leaves reached when applying some combination method. An element of matrix is denoted by $T\_FRF_{t,i}$.
- *D_FRF* is a vector with size I that indicates the confidence assigned by the FRF ensemble to each class i. The matrix elements are obtained from the support for each class in the leaves reached when applying some combination method. Denote an element of this vector as D_FRF_i.

3.2.2. Strategies for fuzzy classifier module in the FRF ensemble

To find the class of an example given with the FRF ensemble, we will define the fuzzy classifier module. The fuzzy classifier module operates on fuzzy trees of the FRF ensemble using one of these two possible strategies:

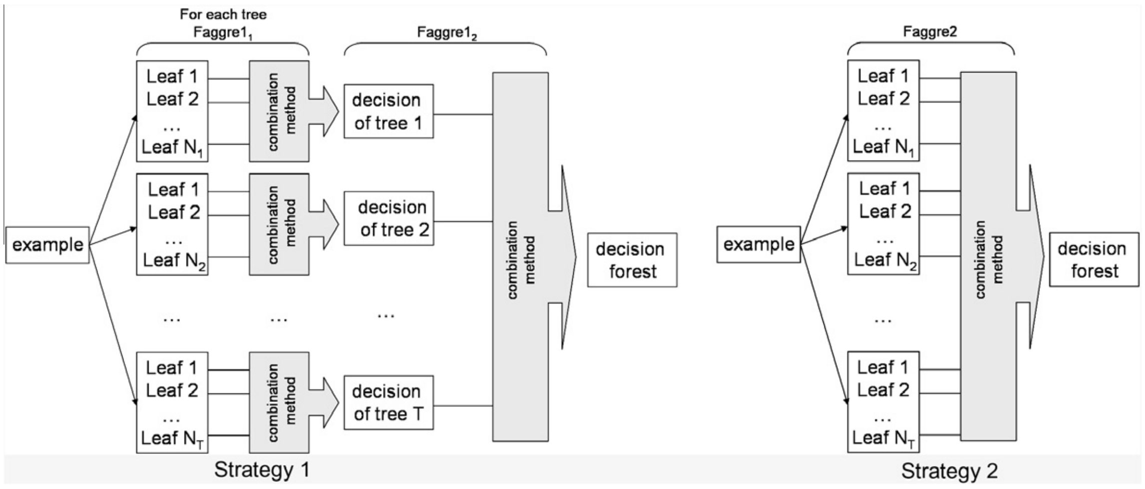

Fig. 1. Strategies for fuzzy classifier module.

Strategy 1: Combining the information from the different leaves reached in each tree to obtain the decision of each individual tree and then applying the same or another combination method to generate the global decision of the FRF ensemble. In order to combine the information of the leaves reached in each tree, the $Faggre1_1$ function is used and the $Faggre1_2$ function is used to combine the outputs obtained with $Faggre1_1$. Fig. 1 shows this strategy.

Strategy 2: Combining the information from all reached leaves from all trees to generate the global decision of the FRF ensemble. We use function $Faggre2$ to combine the information generated by all the leaves. Fig. 1 shows this strategy.

The functions $Faggre1_1, Faggre1_2$ and $Faggre2$ are defined as frequently used combination methods in multiple classifier systems [24,25]. In the next section, we will describe different ways of defining these functions $Faggre1_1$ and $Faggre1_2$, for Strategy 1, and $Faggre2$ for Strategy 2.

In Algorithm 3 we implement Strategy 1.

Algorithm 3. FRF Classification (Strategy 1)

```
FRFclassification(in: e, Fuzzy Random Forest; out: c)
begin
    DecisionsOfTrees(in: e,Fuzzy Random Forest; out: T_FRF)
    DecisionOfForest(in: T_FRF; out: c)
end
DecisionsOfTrees(in: e,Fuzzy Random Forest; out: T_FRF)
begin
  1. Run the example e through each tree to obtain the matrix L_FRF
  2. For each tree t do {For each class i do $T\_FRF_{t,i} = Faggre1_1(t, i, L\_FRF)$}
end
DecisionOfForest(in: T_FRF; out: c)
begin
  1. For each class i do $D\_FRF_i = Faggre1_2(i, T\_FRF)$
  2. The FRF ensemble assigns class c to example e such that $c = \arg\max_{i,i=1,\dots,I}\{D\_FRF_i\}$
end
```

In Algorithm 3, $Faggre1_1$ is used to obtain the matrix T_FRF. In this case, $Faggre1_1$ aggregates the information provided by the leaves reached in a tree. Later, the values obtained in each tree t, will be aggregated by means of the function $Faggre1_2$ to obtain the vector D_FRF. This algorithm takes a target example e and the FRF ensemble, and generates the class value c as a decision of the FRF ensemble.

To implement Strategy 2, the previous Algorithm 3 is simplified so that it does not add the information for each tree, but directly uses the information of all leaves reached by example e in the different trees of the FRF ensemble. Algorithm 4 implements Strategy 2 and uses the example e to classify and the FRF ensemble as target values, and provides the value c as class proposed as decision of the FRF ensemble. $Faggre2$ aggregates the information provided by all leaves reached in the different trees of the FRF ensemble to obtain the vector D_FRF.

Algorithm 4. FRF Classification (Strategy 2)

```
FRFclassification(in: e, Fuzzy Random Forest; out: c)
begin
  1. Run the example $e$ through each tree to obtain the matrix $L\_FRF$
  2. For each class $i$ do $D\_FRF_i = Faggre2(i, L\_FRF)$
  3. The FRF ensemble assigns the class $c$ to example $e$ such that $c = \arg\max_{i,i=1,\dots,I}\{D\_FRF_i\}$
end
```

4. Combination methods in the FRF ensemble

In the previous section, we have shown the general scheme of classification that we use to obtain the final decision of the FRF ensemble. In this section, we describe specific instances of combination methods that we have designed for both strategies.

In all designed methods we will describe functions $Faggre1_1$ and $Faggre1_2$ if it is a method designed for Strategy 1 (Algorithm 3) or only the function $Faggre2$ if it is a method for Strategy 2 (Algorithm 2) also indicating if we use the matrix L_FRF or a transformation of it.

We have divided the several methods implemented in the following groups according to the classification given in Section 2.2:

- *Non-trainable methods*: In this group, we have defined the methods based on the simple majority vote which do not require training during or after the individual training of the ensemble classifiers. This group includes the methods that we have called Simple Majority vote and which, depending on the strategy used for the classification, we will denote as SM1 or SM2 for Strategy 1 or 2, respectively.
- *Trainable methods*: This group includes those methods that require training during or after the individual training of the ensemble classifiers. The methods defined in this group will, through this additional training, obtain the values of certain parameters that act as weightings or weights in the decisions of the various elements of the ensemble (leaves or trees). Within this group we have implemented explicitly data-dependent methods and implicitly data-dependent methods.

– *Explicitly data-dependent methods*: The methods in this sub-group need to learn a parameter which depends on the example to be classified (it depends on the input) and which is common to all the methods of this sub-group. This parameter is the degree of satisfaction with which the example to be classified reaches the various leaves of the trees in the ensemble ($\chi_{t,n}(e)$). Within the sub-group we distinguish:
  - Majority vote Weighted by Leaf for Strategies 1 and 2, (MWL1 and MWL2, respectively). These do not need to learn any other parameter.
  - Majority vote Weighted by Leaf and by Tree for Strategies 1 and 2, (MWLT1 and MWLT2, respectively). These two methods need to learn an additional parameter that indicates the weight of each tree in the decision of the ensemble. This weight is obtained using the OOB dataset.
  - Majority vote Weighted by Leaf and by Local Fusion for Strategies 1 and 2, (MWLFUS1 and MWLFUS2, respectively). These need to learn an additional parameter. Again, it is the parameter for the weight of each tree, and it is obtained by considering the behaviour of each tree with those examples which are similar to the example to be classified (local fusion).
  - Majority vote Weighted by Leaf and by membership Function for Strategies 1 and 2, (MWLF1 and MWLF2, respectively). These need to learn an additional parameter which indicates the weight of each tree in the decision of the ensemble, obtained this time through a membership function that expresses the importance of each tree in proportion to the errors committed with the OOB dataset.
  - Minimum Weighted by Leaf and by membership Function for Strategy 1 (MIWLF1). This method is obtained in the same way as the MWLF1 above but using the minimum instead of the majority vote.

– *Implicitly data-dependent methods*: all the parameters that the methods of this sub-group need to learn do not depend on the example to be classified.
  - Majority vote Weighted by membership Function for Strategies 1 and 2, (MWF1 and MWF2, respectively). It only needs to learn one parameter which indicates the weight of each tree in the decision of the ensemble, which is obtained by a membership function which expresses the importance of each tree in proportion to the errors it committed with the OOB dataset.
  - Minimum vote Weighted by membership Function for Strategies 1 and 2, (MIWF1 and MIWF2, respectively). These two methods are obtained in the same way as methods MWF1 and MWF2, but using the minimum instead of the majority vote.

All the methods presented above are described in detail in the rest of this section.

4.1. Non-trainable methods

Within this group, we define the following methods:

- *Simple Majority vote:* In this combination method, the transformation *TRANS*1 is applied to the matrix *L_FRF* in Step 2 in Algorithms 3 and 4 so that each leaf reached assigns a simple vote to the majority class. We get two versions of this method depending on the strategy used:

Strategy 1 → method SM1
The function $Faggre1_1$ in Algorithm 3 is defined as:

$$Faggre1_1(t,i,L\_FRF) = \begin{cases} 1 & if\ i = \arg\max_{j,j=1,\ldots,I} \left\{ \sum_{n=1}^{N_t} L\_FRF_{t,n,j} \right\} \\ 0 & otherwise \end{cases}$$

In this method, each tree t assigns a simple vote to the most voted class among the N_t reached leaves by example e in the tree.
The function $Faggre1_2$ in Algorithm 3 is defined as:

$$Faggre1_2(i,T\_FRF) = \sum_{t=1}^{T} T\_FRF_{t,i}$$

Strategy 2 → method SM2
For Strategy 2 it is necessary to define the function *Faggre*2 combining information from all leaves reached in the ensemble by example *e*. Thus, the function *Faggre*2 in Algorithm 4 is defined as:

$$Faggre2(i,L\_FRF) = \sum_{t=1}^{T} \sum_{n=1}^{N_t} L\_FRF_{t,n,i}$$

4.2. Trainable explicitly dependent methods

Within this group we define the following methods:

- *Majority vote Weighted by Leaf:* In this combination method, the transformation *TRANS*2 is applied to the matrix *L_FRF* in Step 2 of Algorithms 3 and 4 so that each leaf reached assigns a weighted vote to the majority class. The vote is weighted by the degree of satisfaction with which example *e* reaches the leaf. Again, we have two versions according to the strategy used.

Strategy 1 → method MWL1
The functions $Faggre1_1$ and $Faggre1_2$ are defined as:

$$Faggre1_1(t,i,L\_FRF) = \begin{cases} 1 & if\ i = \arg\max_{j,j=1,\ldots,I} \left\{ \sum_{n=1}^{N_t} L\_FRF_{t,n,j} \right\} \\ 0 & otherwise \end{cases}$$

$$Faggre1_2(i,T\_FRF) = \sum_{t=1}^{T} T\_FRF_{t,i}$$

Strategy 2 → method MWL2
The function *Faggre*2 is defined as:

$$Faggre2(i,L\_FRF) = \sum_{t=1}^{T} \sum_{n=1}^{N_t} L\_FRF_{t,n,i}$$

- *Majority vote Weighted by Leaf and by Tree:* In this method, the transformation *TRANS*2 is applied to matrix *L_FRF* in Step 2 of Algorithms 3 and 4 so that each reached leaf assigns a weighted vote, according to the degree of satisfaction, to the majority class.

In addition, in this method a weight for each tree obtained is introduced by testing each individual tree with the OOB dataset. Let $\overline{p} = (p_1, p_2, \ldots, p_T)$ be the vector with the weights assigned to each tree. Each p_t is obtained as $\frac{N\_success\_OOB_t}{size\_OOB_t}$ where $N\_success\_OOB_t$ is the number of examples classified correctly from the OOB dataset used for testing the tth tree and $size\_OOB_t$ is the total number of examples in this dataset.

Strategy 1 → method MWLT1
The function $Faggre1_1$ is defined as:

$$Faggre1_1(t,i,L\_FRF) = \begin{cases} 1 & \textit{if } i = \arg\max\limits_{j,j=1,\ldots,I} \left\{ \sum\limits_{n=1}^{N_t} L\_FRF_{t,n,j} \right\} \\ 0 & \textit{otherwise} \end{cases}$$

Vector $\overline{p}$ is used in the definition of function $Faggre1_2$:

$$Faggr1_2(i,T\_FRF) = \sum_{t=1}^{T} p_t \cdot T\_FRF_{t,i}$$

Strategy 2 → method MWLT2
The vector of weights $\overline{p}$ is applied to Strategy 2.

$$Faggre2(i,L\_FRF) = \sum_{t=1}^{T} p_t \sum_{n=1}^{N_t} L\_FRF_{t,n,i}$$

- *Majority vote Weighted by Leaf and by Local Fusion:* In this combination method, the transformation *TRANS*2 is applied to the matrix *L_FRF* in Step 2 of Algorithms 3 and 4 so that each reached leaf assigns a weighted vote, again according to the degree of satisfaction, to the majority class.

Strategy 1 → method MWLFUS1
The function $Faggre1_1$ is defined as:

$$Faggre1_1(t,i,L\_FRF) = \begin{cases} 1 & \textit{if } i = \arg\max\limits_{j,j=1,\ldots,I} \left\{ \sum\limits_{n=1}^{N_t} L\_FRF_{t,n,j} \right\} \\ 0 & \textit{otherwise} \end{cases}$$

In addition, for each tree and each example to be classified, a weight is used in the function $Faggre1_2$ which is obtained in the way explained below.

To apply this combination method, first, during the learning of the FRF ensemble, we obtain an additional tree from each tree generated, which we denote error tree. The procedure to construct the error tree associated to *t*th tree is the following:

With the training dataset of the *t*th tree (tds tree$_t$ in Fig. 2) we make a test of the tree. So, we consider the training dataset as the test dataset. With the results of this test we build a new dataset (tds error_tree$_t$ in Fig. 2) with the same data but replacing the class attribute for the binary attribute *error*. The attribute *error* indicates whether that example has been classified correctly or not by the *t*th tree (for example, that binary attribute can take the value 0 if the example has been correctly classified by the tree or 1 if it has been incorrectly classified and therefore it is a mistake made by the tree). With this new dataset a tree is constructed to learn the attribute *error*.

In Fig. 2, tds tree$_t$ is the training dataset of the *t*th tree and it contains examples represented by vectors where

- $e_{j,t}$ is the *j*th example in the training dataset of the *t*th tree;
- $class_{j,t}$ is the value of the attribute class to the *j*th example. This attribute is the classification aim for the FRF ensemble;

and, tds error_tree$_t$ is the training dataset of the error tree associated to the *t*th tree. It contains vectors represented as

- $e_{j,t}$ is the *j*th example in the training dataset of the *t*th tree;
- $error_{j,t}$ is the attribute that acts as class in this dataset. It is a binary attribute with value:
  - 1 if $e_{j,t}$ is incorrectly classified by the *t*th tree;
  - 0 if $e_{j,t}$ is correctly classified by the *t*th tree.

Once the FRF ensemble and the additional error trees have been built, we will obtain a vector $\overline{pl_e} = (pl_{e,1}, pl_{e,2}, \ldots, pl_{e,T})$ for each example *e* to classify and each tree of the FRF ensemble with the weights assigned to each tree with this example (local weights). Each $pl_{e,t}$ is obtained as $pl_{e,t} = \sum_{n=1}^{N_{error_t}} \chi_{error_t,n}(e)\sup_{error_t,n,0}$, $\forall t = 1,\ldots,T$ where the index $error_t$ is the error tree of *t*th tree; N_{error_t} is the number of leaves reached by the example *e* in the error tree $error_t$; $\chi_{error_t,n}(e)$ is the degree of satisfaction with which the example *e* reaches leaf *n* of the error tree $error_t$ and $\sup_{error_t,n,0}$ is the proportion of examples of class 0 (value of binary attribute error = 0) in the leaf *n* of the error tree.

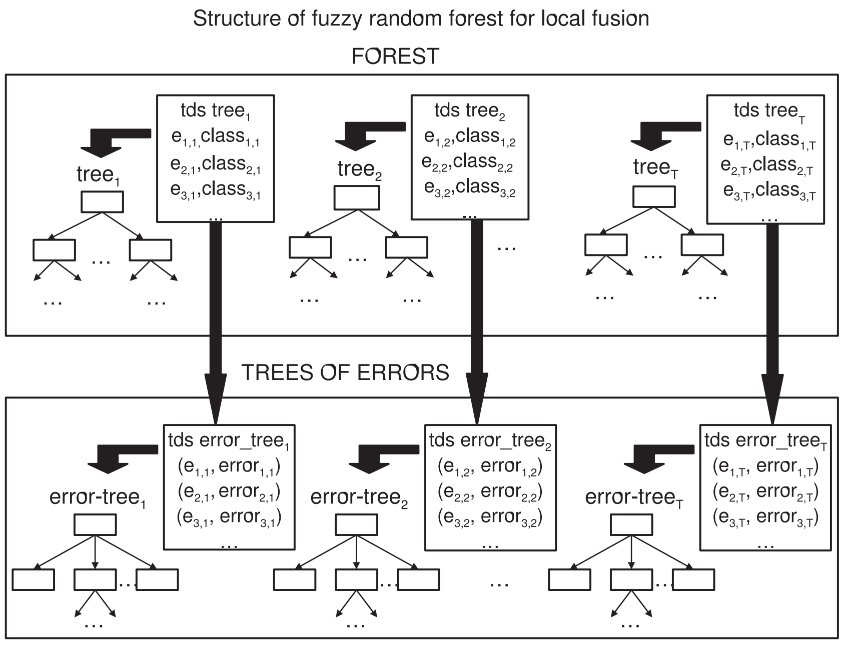
Structure of fuzzy random forest for local fusion
FOREST
tds tree$_1$ e$_{1,1}$,class$_{1,1}$ e$_{2,1}$,class$_{2,1}$ e$_{3,1}$,class$_{3,1}$ …
tree$_1$
tds tree$_2$ e$_{1,2}$,class$_{1,2}$ e$_{2,2}$,class$_{2,2}$ e$_{3,2}$,class$_{3,2}$ …
tree$_2$
tds tree$_T$ e$_{1,T}$,class$_{1,T}$ e$_{2,T}$,class$_{2,T}$ e$_{3,T}$,class$_{3,T}$ …
tree$_T$
TREES OF ERRORS
tds error_tree$_1$ (e$_{1,1}$, error$_{1,1}$) (e$_{2,1}$, error$_{2,1}$) (e$_{3,1}$, error$_{3,1}$) …
error-tree$_1$
tds error_tree$_2$ (e$_{1,2}$, error$_{1,2}$) (e$_{2,2}$, error$_{2,2}$) (e$_{3,2}$, error$_{3,2}$) …
error-tree$_2$
tds error_tree$_T$ (e$_{1,T}$, error$_{1,T}$) (e$_{2,T}$, error$_{2,T}$) (e$_{3,T}$, error$_{3,T}$) …
error-tree$_T$

Fig. 2. Classification using local fusion.

The key idea that we want to capture with this method is the use of a local weight or a local fusion mechanism [5]. Given a new example, we first evaluate the performance of the tree with those examples (in the training dataset) which are similar to it. Similar examples are those which belong to the leaf node of the error tree which has activated the example to the greatest degree. Then, we associate a weight to the decision of that tree on the basis of its performance with those examples.

Finally, the function $Faggre1_2$ is defined by weighting the decision of each tree with the weight obtained specifically for example e and tree t:

$$Faggre1_2(i, T\_FRF) = \sum_{t=1}^{T} pl_{e,t} \cdot T\_FRF_{t,i}$$

Strategy 2 → method MWLFUS2
This method uses the vector of weights $\overline{pl}_e$ applied to Strategy 2.

$$Faggre2(i, L\_FRF) = \sum_{t=1}^{T} pl_{e,t} \sum_{n=1}^{N_t} L\_FRF_{t,n,i}$$

- *Majority vote Weighted by Leaf and by membership Function:*
  In this combination method, the transformation *TRANS*2 is applied to the matrix *L_FRF* in Step 2 of Algorithms 3 and 4 so that each reached leaf assigns a weighted vote, again according to the degree of satisfaction, to the majority class.

Strategy 1 → method MWLF1
The function $Faggre1_1$ is defined as:

$$Faggre1_1(t, i, L\_FRF) = \begin{cases} 1 & if\ i = \arg\max_{j,j=1,\dots,I} \left\{ \sum_{n=1}^{N_t} L\_FRF_{t,n,j} \right\} \\ 0 & otherwise \end{cases}$$

In this method, the function $Faggre1_2$ weights the decision of each tree of the FRF ensemble using the membership function defined by $\mu_{pond}(x)$:

$$\mu_{pond}(x) = \begin{cases} 1 & 0 \leqslant x \leqslant (pmin + marg) \\ \frac{(pmax+marg)-x}{(pmax-pmin)} & (pmin + marg) \leqslant x \leqslant (pmax + marg) \\ 0 & (pmax + marg) \leqslant x \end{cases}$$

where

– *pmax* is the maximum rate of errors in the trees of the FRF ensemble ($pmax = \max_{t=1,\ldots,T}\left\{\frac{errors_{(OOB_t)}}{size_{(OOB_t)}}\right\}$). The rate of errors in a tree *t* is obtained as $\frac{errors_{(OOB_t)}}{size_{(OOB_t)}}$ where $errors_{(OOB_t)}$ is the number of classification errors of the tree *t* (using the OOB_t dataset as test set), and $size_{(OOB_t)}$ is the cardinal of the OOB_t dataset. As we indicated above, the OOB_t examples are not used to build the tree *t* and they constitute an independent sample to test tree *t*. So we can measure the goodness of a tree *t* as the number of errors when classifying the set of examples OOB_t;
– *pmin* is the minimum rate of errors in the trees of the FRF ensemble; and
– $marg = \frac{pmax-pmin}{4}$.

With this membership function, all trees have a weight greater than 0 in the decision of the FRF ensemble. The weight is decreased when the rate of errors increases so that trees with a minimum rate of error will have a weight equal to 1.
So, the function $Faggre1_2$ is defined as:

$$Faggre1_2(i, T\_FRF) = \sum_{t=1}^{T} \mu_{pond}\left(\frac{errors_{(OOB_t)}}{size_{(OOB_t)}}\right) \cdot T\_FRF_{t,i}$$

Strategy 2 → method MWLF2
In this method the function *Faggre*2 is defined as:

$$Faggre2(i, L\_FRF) = \sum_{t=1}^{T} \mu_{pond}\left(\frac{errors_{(OOB_t)}}{size_{(OOB_t)}}\right) \cdot \sum_{n=1}^{N_t} L\_FRF_{t,n,i}$$

- *Minimum Weighted by Leaf and by membership Function:*
  In this combination method, the transformation *TRANS*3 is applied to the matrix *L_FRF* in Step 2 of Algorithm 3.

Strategy 1 → method MIWLF1
The function $Faggre1_1$ is defined as:

$$Faggre1_1(t, i, L\_FRF) = \begin{cases} 1 & \textit{if } i = \arg\max_{j,j=1,\ldots,I}\{min(L\_FRF_{t,1,j}, L\_FRF_{t,2,j}, \ldots, L\_FRF_{t,N_t,j})\} \\ 0 & \textit{otherwise} \end{cases}$$

The function $Faggre1_2$ incorporates the weighting defined by the previous fuzzy membership function for each tree:

$$Faggre1_2(i, T\_FRF) = \sum_{t=1}^{T} \mu_{pond}\left(\frac{errors_{(OOB_t)}}{size_{(OOB_t)}}\right) \cdot T\_FRF_{t,i}$$

4.3. Trainable implicitly dependent methods

Within this group we define the following methods:

- *Majority vote Weighted by membership Function:* In this combination method, the transformation *TRANS*1 is applied to the matrix *L_FRF* in the Step 2 in Algorithms 3 and 4 so that each reached leaf assigns a simple vote to the majority class.

Strategy 1 → method MWF1
The function $Faggre1_1$ is defined as:

$$Faggre1_1(t, i, L\_FRF) = \begin{cases} 1 & \textit{if } i = \arg\max_{j,j=1,\ldots,I}\left\{\sum_{n=1}^{N_t} L\_FRF_{t,n,j}\right\} \\ 0 & \textit{otherwise} \end{cases}$$

The function $Faggre1_2$ incorporates the weighting defined by the previous fuzzy membership function for each tree:

$$Faggre_2(i, T\_FRF) = \sum_{t=1}^{T} \mu_{pond}\left(\frac{errors_{(OOB_t)}}{size_{(OOB_t)}}\right) \cdot T\_FRF_{t,i}$$

Strategy 2 → method MWF2
The function *Faggre*2 incorporates the weighting defined by the previous fuzzy membership function:

$$Faggre_2(i, L\_FRF) = \sum_{t=1}^{T} \mu_{pond}\left(\frac{errors_{(OOB_t)}}{size_{(OOB_t)}}\right) \cdot \sum_{n=1}^{N_t} L\_FRF_{t,n,i}$$

- *Minimum Weighted by membership Function:* In this combination method, no transformation is applied to matrix *L_FRF* in Step 2 of Algorithm 3.

Strategy 1 → method MIWF1
The function $Faggre1_1$ is defined as:

$$Faggre1_1(t, i, L\_FRF) = \begin{cases} 1 & \textit{if } i = \arg\max_{j,j=1,\ldots,I} \{min(L\_FRF_{t,1,j}, L\_FRF_{t,2,j}, \ldots, L\_FRF_{t,N_t,j})\} \\ 0 & \textit{otherwise} \end{cases}$$

The function $Faggre1_2$ incorporates the weighting defined by the previous fuzzy membership function for each tree:

$$Faggre1_2(i, T\_FRF) = \sum_{t=1}^{T} \mu_{pond}\left(\frac{errors_{(OOB_t)}}{size_{(OOB_t)}}\right) \cdot T\_FRF_{t,i}$$

5. Experiments and results

In this section, we describe several computational results, which show the accuracy of the proposed FRF ensemble. The experiments are grouped as follows:

- The experiments of Section 5.3 are designed to measure the behaviour and stability of the FRF ensemble with imperfect data and noisy data. In other words, we want to test the FRF behaviour against dataset that contain missing values, values provided by fuzzy sets (fuzzy values), noise in the class or outlier examples. We therefore form two groups of experiments:
  – FRF behaviour with imperfect data:
    * Missing values and
    * Fuzzy values.

  – FRF behaviour with noise:
    * Noise in the class and
    * Outlier examples.
- The experiments of Section 5.4 are designed to compare the FRF ensemble with other classifiers and ensembles.
  – First, we compare the FRF ensemble with other ensembles. These ensembles are formed with same base classifier as the FRF ensemble (a fuzzy decision tree). We also use Breiman's Random Forest.
  – Second, we compare and baseline the operation of the FRF ensemble with other classifiers and ensembles found in the literature.

Table 1
Datasets.

Dataset	Abbr	$\lvert E\rvert$	$\lvert M\rvert$	I
Appendicitis	APE	106	7	2
Wisconsin breast C.	BCW	683	9	2
BUPA liver disorders	BLD	345	6	2
Contraceptive method	CMC	1473	9	3
German credit	GER	1000	24	2
Glass	GLA	214	9	7
Statlog heart disease	HEA	270	13	2
Ionosphere	ION	351	34	2
Iris plants	IRP	150	4	3
Pima Indian diabetes	PID	532	7	2
Pima Indian diabetes	PIM	768	8	2
Image segmentation	SEG	2130	19	7
Attitude smoking	SMO	2855	8	3
Thyroid disease	THY	7200	21	3
Vehicle	VEH	846	18	4
Wine	WIN	178	13	3

Table 2
Testing accuracies of the FRF ensemble for different percentages of missing values.

Dataset	Introducing missing values			
	Without	5%	15%	30%
		% Decrease average accuracy		
APE	$91.13_{(9.70)\ \mathrm{MIWF1/MIWLF1}}$	$0.82_{\mathrm{MIWF1/MIWLF1}}$	$1.03_{\mathrm{MWLT2/MWLF2}}$	$0.21_{\mathrm{MIWF1/MIWLF1}}$
BCW	$97.31_{(1.76)\ \mathrm{MWLT1/MWLT2}}$	$0.12_{\mathrm{MWL2/MWLT2}}$	0.79_{MWLT2}	2.92_{MWLT2}
GER	$76.68_{(3.97)\ \mathrm{MWLF2}}$	0.70_{MWLT2}	3.86_{MWLFUS2}	5.16_{MWLFUS2}
GLA	$77.66_{(7.36)\ \mathrm{MWLT2}}$	6.62_{MWLFUS2}	10.95_{MWL2}	17.20_{MWLT2}
ION	$96.41_{(2.89)\ \mathrm{MIWLF1}}$	$0.94_{\mathrm{MIWF1/MIWLF1}}$	2.66_{MWLFUS2}	$6.09_{\mathrm{MIWF1/MIWLF1}}$
IRP	$97.33_{(3.93)\ **}$	$1.23_{\mathrm{SM1/SM2//MWF1/MWF2}}$	4.11_{MWF1}	16.71_{MWL2}
PIM	$77.14_{(4.88)\ \mathrm{MWF1}}$	0.82_{MWF1}	2.57_{MWF1}	7.47_{MWLF2}
WIN	$97.87_{(2.85)\ \mathrm{MWLF1}}$	$4.41_{\mathrm{MIWF1/MIWLF1}}$	6.27_{MWL2}	$14.21_{\mathrm{MIWF1/MIWLF1}}$

5.1. Datasets and parameters for the FRF ensemble

To obtain these results we have used several datasets from the UCI repository [2], whose characteristics are shown in Table 1. It shows the number of examples ($|E|$), the number of attributes ($|M|$) and the number of classes (I) for each dataset. "Abbr" indicates the abbreviation of the dataset used in the experiments.

Finally, we use the FRF ensemble with size $T \in (100, 150)$ trees except for the experiment of Section 5.4.1, which will be shown in Table 7. The number of attributes chosen at random at a given node is $\log_2(|\cdot| + 1)$, where $|\cdot|$ is the number of available attributes at that node, and each tree of the FRF ensemble is constructed to the maximum size (node pure or set of available attributes is empty) and without pruning.

5.2. Validating the experimental results by non-parametric tests

After the experimental results have been shown, we make an analysis of them in each subsection using statistical techniques. Following the methodology of [16] we use non-parametric tests.

We use the Wilcoxon signed-rank test to compare two methods. This test is a non-parametric statistical procedure for performing pairwise comparison between two methods. This is analogous with the paired *t*-test in non-parametric statistical procedures; therefore, it is a pairwise test that aims to detect significant differences between two sample means, that is, the behaviour of two methods.

When we compare multiple methods, we use the Friedman test and the Benjamin–Hochberger procedure [4] as post-hoc test (this last procedure is more powerful than Bonferroni–Dunn test, Holm test and Hochberger procedure). The Friedman test is a non-parametric test equivalent to the repeated-measures ANOVA. Under the null-hypothesis, it states that the methods are equivalent, so a rejection of this hypothesis implies the existence of differences in the performance of all the methods studied. After this, the Benjamin–Hochberger procedure is used as a post-hoc test to find whether the control or proposed methods show statistical differences with regard to the other methods in the comparison.

5.3. Behaviour and stability of the FRF ensemble with imperfect data and noise

5.3.1. Management of imperfect data

To introduce an $a\%$ of imperfect values in a dataset of $|E|$ examples, each of which has $|M|$ attributes (excluding the class attribute), we select randomly $a\% \cdot |E| \cdot |M|$ values of the dataset uniformly distributed among all the attributes. For each value, corresponding to an example and to an attribute, we modify the value. Imperfect data were introduced to both the training and testing datasets.

We divided this test in three experiments:

- In the first experiment, we run the FRF ensemble on datasets in which we have inserted missing values, in both numerical and nominal attributes.
- In the second experiment, we run the FRF ensemble on datasets in which we have inserted fuzzy values in the numerical attributes. These fuzzy values correspond to the different fuzzy sets of the fuzzy partition for each numeric attribute of the dataset.
- In the third experiment, we inserted tanto missing values como fuzzy values on datasets.

When the value of a numerical attribute of an example from the dataset is chosen to be replaced by a fuzzy value, it is done as follows: since the numerical attribute is partitioned in a fuzzy partition, the value of the attribute will belong, with associated degrees of membership, to one or two fuzzy sets of the partition. We substitute the value of the attribute of that example for the fuzzy set with which the greatest degree of membership was obtained. The percentages of imperfect data

Table 3
Testing accuracies of the FRF ensemble for different percentages of fuzzy values.

Dataset	Introducing fuzzy values			
	Without	5%	15%	30%
		% Decrease average accuracy		
APE	$91.13_{(9.70)\ \mathrm{MIWF1/MIWLF1}}$	$-0.21_{\mathrm{MIWF1/MIWLF1}}$	$0.82_{\mathrm{MIWF1/MIWLF1MWLF1/MWLF2}}$	$0.21_{\mathrm{MIWF1/MIWLF1}}$
BCW	$97.31_{(1.76)\ \mathrm{MWLT1/MWLT2}}$	$0.42_{\mathrm{MWLT1/MWLT2MWLFUS1/MWLFUS2}}$	$0.28_{\mathrm{MWLT1/MWLT2}}$	$0.12_{\mathrm{MWLT1/MWLT2}}$
GER	$76.68_{(3.97)\ \mathrm{MWLF2}}$	$0.08_{\mathrm{MIWF1/MIWLF1}}$	-0.29_{MIWLF1}	-0.26_{MWLF2}
GLA	$77.66_{(7.36)\ \mathrm{MWLT2}}$	-1.08_{MWLF2}	0.00_{SM2}	1.31_{MWF2}
ION	$96.41_{(2.89)\ \mathrm{MIWLF1}}$	0.53_{MIWLF1}	0.46_{MWLF2}	$0.94_{\mathrm{MIWF1/MIWLF1}}$
IRP	$97.33_{(2.14)\ **}$	0.00 **	0.00_{**}	0.00 **
PIM	$77.14_{(4.88)\ \mathrm{MWF1}}$	0.52_{MWF1}	0.17_{MWF1}	1.49_{SM1}
WIN	$97.87_{(2.85)\ \mathrm{MWLF1}}$	$-0.29_{\mathrm{MWL2/MWLT2/MWLFUS2/MWLF2}}$	$-0.51_{\mathrm{MWLF1/MWLF2}}$	0.75_{MWF2}

Table 4
Testing accuracies of the FRF ensemble for different percentages of missing and fuzzy values.

Dataset	Introducing missing values and fuzzy values			
	Without	5%	15%	30%
		% Decrease average accuracy		
APE	$91.13_{(9.70)\ \mathrm{MIWF1/MIWLF1}}$	$0.00_{\mathrm{MIWF1/MIWLF1}}$	$1.03_{\mathrm{MIWF1/MIWLF1}}$	0.21_{MWF2}
BCW	$97.31_{(1.76)\ \mathrm{MWLT1/MWLT2}}$	0.76_{MWLFUS1}	$0.31_{\mathrm{MWL1/MWLT1}}$	0.67_{MWF1}
GER	$76.68_{(3.97)\ \mathrm{MWLF2}}$	0.76_{MIWLF1}	$1.10_{\mathrm{MIWF1/MIWLF1}}$	$2.50_{\mathrm{MIWF1/MIWLF1}}$
GLA	$77.66_{(7.36)\ \mathrm{MWLT2}}$	0.59_{MWL2}	5.90_{MWLF2}	10.11_{MWLFUS2}
ION	$96.41_{(2.89)\ \mathrm{MIWLF1}}$	1.18_{MWLFUS2}	$2.83_{\mathrm{MIWF1/MIWLF1}}$	$3.25_{\mathrm{MIWF1/MIWLF1}}$
IRP	$97.33_{(2.14)\ **}$	1.91 **	$4.11_{\mathrm{MWL1/MWLT1/MWLFUS1/MWLF1}}$	6.99_{MWLF2}
PIM	$77.14_{(4.88)\ \mathrm{MWF1}}$	1.11_{MWF1}	1.19_{MWLT1}	1.36_{MWF1}
WIN	$97.87_{(2.85)\ \mathrm{MWLF1}}$	0.29_{MWLFUS1}	4.89_{MWL2}	$7.07_{\mathrm{MIWF1/MIWLF1}}$

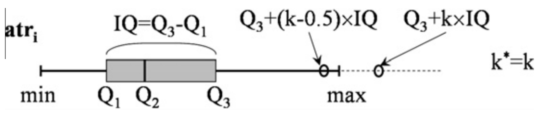

Fig. 3. Distribution of attribute Atr_i and value of k^*.

inserted in the datasets were 5%, 15%, and 30% in each of the three experiments[1]. In the third experiment, the percentage was divided into equal parts of missing values and fuzzy values.

In these experiments, a 10-fold cross validation is independently performed five times using different partitions of the dataset (5×10-fold cross validation) and we show the percentage of classification average accuracy of the FRF ensemble (mean and standard deviation) in dataset without imperfect data and decrease in the percentage of classification average accuracy of the FRF ensemble in the dataset with imperfect data, together with the combination methods of the FRF ensemble which obtains these values (the symbol "**" indicates that there are more than four methods that obtain that average value). The decrease in the percentage of classification average accuracy, which is shown in Tables 2–4, is computed as, $\%decrease\ accuracy = 100 \cdot \frac{(CP(original)-CP(imperfect))}{CP(original)}$ where *CP(imperfect)* is the classification average accuracy for the dataset with imperfect data, and *CP(original)* is the one for the original data.

In Tables 2–4 it can be observed that the FRF ensemble presents a very stable behaviour in the presence of a significant amount of imperfect data.

5.3.2. *Effect of noise*

In this test, we analyze the effect that noise can cause in the FRF ensemble. We divided this test in two experiments. In the first, we ran the FRF ensemble on datasets in which we introduced outlier examples. In the second experiment, we ran the FRF ensemble on datasets in which we inserted data with noise in the class attribute.

5.3.2.1. *Introducing outliers in datasets.*

One way to identify outliers is with the quartile method. This method uses the lower quartile or 25th percentile (Q_1) and the upper quartile or 75th percentile (Q_3) of each attribute of the dataset (the Q_2 quartile

[1] In order to obtain these datasets with imperfect data we used the NIP 1.5 tool [12].

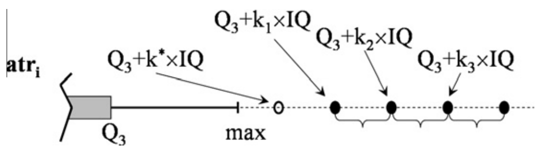

Fig. 4. Three possible values for generating outliers.

Table 5
Effect for different types of outliers on the FRF ensemble.

Dataset	Outliers			
	Without	Obtained with k_1	Obtained with k_2	Obtained with k_3
		% Increase average error		
BCW	$97.30_{(1.48)}$ MWLFUS1/MWLFUS2	-1.11 MWF1/MWLF1	-5.19 MWLFUS1/MWLFUS2	-3.70 MWLFUS1/MWLFUS2
BLD	$72.97_{(5.19)}$ MWF2	-3.22 MWF2	-2.70 MWF2	1.59 MWF2
CMC	$53.62_{(2.00)}$ MWLF1	0.19 MWLT1	0.30 MWLT1	-0.11 MWLF1
GLA	$78.38_{(6.11)}$ MWLT2	1.57 MWLT2	3.75 MWLT2	1.06 MWLT2
HEA	$82.87_{(5.29)}$ MWLT1	-4.32 MWLT1	-1.11 MWLF2	-4.32 MWLT1
ION	$94.66_{(2.19)}$ SM1	6.93 MWF1	6.74 SM1	5.43 SM1
IRP	$97.33_{(2.14)}$ **	0.00**	0.00**	0.00**
PID	$79.61_{(3.27)}$ MWF2	1.62 MWF2	0.49 MIWLF1	2.11 MWF2
PIM	$76.53_{(3.86)}$ MWF1	3.07 MWF1	1.79 SM1	-0.13 SM1
SEG	$97.19_{(0.48)}$ MWF1	2.85 MWF1	3.91 MWF1	3.20 MIWLF1
SMO	$69.54_{(1.97)}$ MIWF1/MIWLF1	0.03 MIWF1/MIWLF1	0.03 MIWF1/MIWLF1	0.03 MIWF1/MIWLF1
VEH	$75.18_{(1.91)}$ MWLF2	0.60 MWF1	1.93 MWF2	1.33 MWF2
WIN	$97.48_{(3.23)}$ MWLF1	17.06 MWLF1	11.51 MWLF1	11.51 MWLF1

corresponds to the median and *min* and *max* correspond to the lowest and highest value of each attribute, respectively). We will use this quartile method to generate and insert examples with outlier values in the different datasets.

We will take an outlier value as that which is greater than $Q_3 + k \cdot IQ$ where k is a given positive constant and IQ is the inter-quartile range. Thus, the datasets with outliers were obtained in the following way:

1. We selected a numerical attribute (Atr_i) for each dataset.
2. For each dataset and its selected attribute Atr_i, we calculate $k^* = min\{k/Q_3 + (k - 0.5) \cdot IQ \leqslant (\text{max value } Atr_i \text{ in } E) \leqslant Q_3 + k \cdot IQ\}$ with E being the set of examples from the datasets, k takes values in $\{0.5, 1, 1.5, 2, 2.5, \ldots\}$ and $IQ = Q_3 - Q_1$ (inter-quartile range), Q_1 lower quartile (25th percentile), Q_3 upper quartile (75th percentile), respectively, of the attribute Atr_i (see Fig. 3).
3. For each dataset, we select a 1% of examples.
4. We define $k_1 = k^* + 0.5,\ k_2 = k^* + 1$ and $k_3 = k^* + 1.5$.
5. For each example selected, we modified the value of the numerical attribute Atr_i by replacing it by a value chosen randomly from interval $[Q_3 + k_i \cdot IQ, Q_3 + (k_i + 0.5) \cdot IQ]$ for $i = 1, 2, 3$. As can be observed (see Fig. 4), we obtain three possible values for each replacement, depending on k_1, k_2 and k_3. Hence, we will obtain three datasets with outliers corresponding to k_1, k_2 and k_3 for each original dataset. This was done only for the training dataset.

We carried out three experiments, one for each dataset obtained in the previous process with the selected k_1, k_2 and k_3. The experiments were done with a 4×5-fold cross validation. Table 5 shows the percentage of classification average accuracy values (mean and standard deviation) for datasets without outliers and the percentage increase in the classification

Table 6
Comparison of tree-based ensembles and classifiers with noise data.

Dataset	% Increase error Hamza et al. [18]	Best classification method Hamza et al. [18]	% Increase error (FRF)	Best combination method (FRF)
BCW	54.84	ST-NLC-G	4.54	MWLFUS2
BLD	16.98	BA-NLC-G	6.78	MWLFUS2
CMC	-6.23	ST-NLC-G	0.57	MWLFUS2
HEA	-2.08	RF	-2.44	MIWF1
PID	5.74	RF	4.42	MWLFUS2
SEG	115.56	RF	81.36	MWF2
SMO	3.07	BA-NLC-G	0.02	MWLT2,MWLFUS2
THY	23.08	RF	18.19	MWF2
VEH	-0.76	ST-NLC-G	-0.95	MIWF1

average error between the original data and data with outliers. In addition, the combination method that obtains these values is indicated (the symbol "**" indicates that there are more than four combination methods that obtain that value). The increase in the percentage of classification average error, shown in the Table 5, is computed as $\%increase\ error = 100 \cdot \frac{(CE(with\_outliers) - CE(original))}{CE(original)}$ where *CE(with_outliers)* is the classification average error for the dataset with outliers, and *CE(original)* is the one for the original data.

When we perform the Friedman non-parametric statistical test to compare the average accuracy of these four samples we find no significant differences between them with a 95% confidence level. From these results, we can conclude that introducing outliers further away from the sample causes the FRF ensemble to behave just as if it had no outliers.

5.3.2.2. Introducing noise data in the class attribute. We compared the FRF ensemble with the best technique reported in [18] for the same experiment. The best technique is defined as the one with the lowest percentage increase in the classification average error between the original dataset and dataset with noise, with a 10-fold cross validation.

The datasets with noise were obtained in the following way: with probability 10%, we modified the value of the class attribute by replacing it with a value chosen uniformly at random from its other possible values. This was only for the training dataset. Again, noise was introduced into the training datasets via the NIP 1.5 tool [12]. The increase in the percentage of classification average error, which is shown in the Table 6, is computed as $\%increase\ error = 100 \cdot \frac{(CE(noise) - CE(original))}{CE(original)}$ where $CE(noise)$ is the classification error for the dataset with noise, and $CE(original)$ is that for the original data.

The results can be seen in Table 6. Again, using the Wilcoxon test to compare the results of [18] and the FRF ensemble, we obtain significant differences at 97.3%. With these results, the FRF ensemble has a good behaviour when we introduce noise in the class attribute with an increase of error smaller than in [18].

5.4. Comparing the FRF ensemble with other classifiers and ensembles

5.4.1. Comparative of the FRF ensemble and other ensembles using the same base classifier

This subsection summarizes a series of experiments performed to observe the effectiveness of the FRF ensemble when compared with the base classifier and several ensembles built with this base classifier: (1) the base classifier, (2) a boosting based ensemble, (3) a bagging based ensemble, and (4) the FRF ensemble. We also compare the results of the FRF ensemble with the obtained with Breiman's Random Forest (RF) [17]. We should note that, excluding RF, all ensembles have been built using a fuzzy decision tree as base classifier. The execution of each experiment was done with the same parameters. In this experiment we have made a 4×5-fold cross validation. Table 7 shows the results obtained, indicating the percentage of classification average accuracy (mean and standard deviation).

The results obtained in this experiment clearly show that, for these datasets, the ensembles are always better than the individual classifier. It is also shown that the FRF ensemble is the ensemble that consistently generates the best results. In most cases bagging is better than boosting. When we perform the statistic test on these results, we first apply the Friedman test, obtaining a rejection of the null-hypothesis with a 99.9% confidence level. That is, it accepts that there are significant differences. When we perform the post-hoc test, we obtain that the FRF ensemble has significant differences with methods RF, fuzzy decision tree (FT), boosting and bagging with a confidence level of 95.98%, where the FRF ensemble is the best method. For other methods we obtain the following: with a 99.9% confidence level it is concluded that RF, FT and boosting are significantly different, where RF is the best of them; and with a 99.7% confidence level it is concluded that bagging, FT and boosting are significantly different.

5.4.2. Comparative with other methods of the literature

In this subsection, we compare and baseline the operation of the FRF ensemble with other classifiers and ensembles found in the literature. In each case we will say how the comparison has been made.

Table 7
Testing average accuracies of the FRF ensemble with other ensembles with same base classifier.

Dataset	Size RF	RF	fuzzy tree (FT)	Size ensembles	Boosting with FT	Bagging with FT	FRF ensemble
BCW	125	$97.07_{(1.89)}$	$95.50_{(1.75)}$	125	$94.51_{(1.51)}$	$95.68_{(1.60)}$	$97.30_{(1.48)}$
BLD	200	$72.68_{(6.32)}$	$65.14_{(5.75)}$	200	$65.79_{(6.23)}$	$71.88_{(5.89)}$	$72.97_{(5.19)}$
CMC	120	$51.41_{(2.61)}$	$47.17_{(2.62)}$	120	$49.08_{(2.99)}$	$51.49_{(2.04)}$	$53.62_{(2.00)}$
GLA	120	$78.85_{(6.05)}$	$72.43_{(8.34)}$	50	$74.89_{(6.29)}$	$76.74_{(5.72)}$	$78.38_{(6.11)}$
HEA	120	$81.48_{(4.13)}$	$74.26_{(6.58)}$	120	$77.13_{(4.63)}$	$81.02_{(5.13)}$	$82.87_{(5.29)}$
ION	175	$93.45_{(2.25)}$	$92.59_{(3.29)}$	175	$94.09_{(3.59)}$	$93.25_{(3.12)}$	$94.66_{(2.19)}$
IRP	120	$95.33_{(1.74)}$	$97.00_{(2.08)}$	120	$96.67_{(2.53)}$	$96.67_{(2.53)}$	$97.33_{(2.14)}$
PID	125	$76.41_{(2.12)}$	$71.85_{(3.95)}$	50	$70.54_{(3.69)}$	$78.05_{(3.19)}$	$79.61_{(3.27)}$
PIM	150	$75.26_{(3.51)}$	$67.55_{(4.53)}$	150	$66.18_{(3.64)}$	$73.63_{(2.96)}$	$76.53_{(3.86)}$
SEG	140	$97.85_{(0.59)}$	$95.54_{(1.06)}$	140	$96.54_{(0.77)}$	$97.19_{(0.66)}$	$97.19_{(0.48)}$
SMO	100	$61.63_{(0.82)}$	$55.29_{(2.42)}$	75	$56.36_{(1.48)}$	$69.50_{(1.91)}$	$69.54_{(1.97)}$
THY	150	$99.67_{(0.10)}$	$96.13_{(0.53)}$	150	$96.26_{(0.62)}$	$98.25_{(0.38)}$	$99.17_{(0.22)}$
VEH	200	$76.27_{(2.73)}$	$67.96_{(3.94)}$	200	$70.06_{(3.64)}$	$74.41_{(3.04)}$	$75.18_{(1.91)}$
WIN	150	$98.03_{(1.93)}$	$97.19_{(2.85)}$	150	$97.20_{(3.45)}$	$97.06_{(3.36)}$	$97.48_{(3.23)}$

Table 8
Comparative accuracies of the FRF ensemble and other classifiers.

Dataset	Technique									
	FRF	GRA	CIGRA	MLP	C4.5	RBF	Bayes	Cart	GBLM	Fuzzy D.Tree
APE	$91.04_{(10.12)}$ MIWF1/MIWLF1	86.00	88.70	85.80	84.90	80.20	83.00	84.90	–	–
BCW	$97.14_{(1.78)}$ MWLT1/MWLT2	96.20	96.80	96.50	94.70	96.60	96.40	94.40	96.70	96.80
GER	$76.41_{(3.96)}$ MWLF2	73.00	74.20	71.60	73.50	75.70	70.40	73.90	–	–
GLA	$76.82_{(7.85)}$ MWLT2	57.40	63.20	68.70	65.80	46.70	71.80	63.60	65.40	66.00
ION	$96.13_{(2.99)}$ MIWLF1	88.50	92.60	92.00	90.90	94.60	85.50	89.50	–	86.50
IRP	$97.33_{(4.58)}$ **	95.70	96.10	96.00	94.00	98.00	94.70	92.00	94.70	96.10
PIM	$76.70_{(4.34)}$ MWF1	74.90	76.20	75.80	72.70	75.70	72.20	74.70	75.80	73.10
WIN	$97.64_{(3.19)}$ MWLF1	93.30	96.20	98.30	93.30	94.90	94.40	87.60	95.10	91.20

Table 9
Comparison with tree-based ensembles.

	Dataset								
	BCW	BLD	CMC	HEA	PID	SEG	SMO	THY	VEH
Best classification error [18]	2.64	25.80	47.39	17.78	22.93	1.60	34.05	0.28	23.40
Best classification error method [18]	RF	RF	RF	RF	RF	RF	BO(100)WLC-E	BO(100)WLC-E	BO(250)WLC-G
Best classification error – FRF	2.49	24.67	46.57	14.44	19.35	2.55	30.47	0.78	23.17
Best classification error combination method	MWLT1/MWLT2/ MWLFUS1/ MWLFUS2	MWLFUS1	MWLF1	MIWF1	MWLFUS2	MWF2	MWLT2 MWLFUS2	MIWF1/ MIWLF1	MWF2

5.4.2.1. Comparative study with other classifiers. We have compared the results of the FRF ensemble with other classifiers, taking the results reported in [20] which compare the classifiers GRA-based (grey relational analysis) and CIGRA-Based (Choquet integral-based GRA) with other well known classification methods, including the MLP (multi-layer perceptron), the C4.5 decision tree, radial basis function (RBF), the naive Bayes, the Cart decision tree, the hybrid fuzzy genetic-based machine learning algorithm (GBML) and a fuzzy decision tree.

To examine the generalization ability of the FRF ensemble, we have made a 10×10-fold cross validation. Again, we show the percentage of classification average accuracy for all methods and in addition the standard deviation for the FRF ensemble together with the combination methods of the FRF ensemble which obtain these values. The results can be seen in the Table 8.

When we make the statistical analysis for these results, we first apply the Friedman's test getting a rejection of the null-hypothesis with a 99.6% confidence level. That is, we accept that there are significant differences. When we make the post-hoc analysis, we obtain that the FRF ensemble has significant differences with the methods GRA, CIGRA, MLP, C4.5, RBF, Bayes, Cart and Fuzzy D. Tree with a 98.2% confidence level and with GBLM with a 96.9% confidence level, and the FRF ensemble is the best method. We conclude that the FRF ensemble is an effective classifier and that it exhibits very good performance.

5.4.2.2. Comparative study with other ensembles. In Ref. [18] we find a comparative study of the best tree-based ensembles. We will compare the results of the FRF ensemble with those reported in that work. A 10-fold cross validation is made. Then, we very briefly describe the tree-based ensembles used in that work with which we make the comparison. The ensembles used are:

1. Single Tree with pruning (CART).
2. Bagging with 100 trees (CART).
3. *RF*: Random Forest with 100 trees (number of attributes chosen at random at a given node is $log_2(|M| + 1)$ where M is the set of attributes).
4. *BO*: Boosting (arcing) with 100 and 250 trees (CART). Split Criteria – G: Gini, E: Entropy, T: Twoing, WLC: With Linear Combination, NLC: No Linear Combination.

The results can be seen in Table 9. On comparing the FRF ensemble with the best proposed ensemble in [18], with a 95.2% confidence level, there are significant differences between the two methods, with the FRF ensemble being the best.

6. Summary

In this paper, we present an ensemble based on fuzzy decision trees called FRF ensemble. We realize a hybridization of the techniques of random forest and fuzzy trees for training. The proposed ensemble has the advantages of imperfect data management, of being robust to noise and of having a good degree of classification with relatively small sizes of ensembles.

We have defined various methods to combine the outputs of base classifiers of the FRF ensemble. These methods are based on the combination methods used frequently in the literature to obtain the final decision in ensembles. Hence we have defined:

- *Non-trainable methods*: Within this group there are methods based on simple majority vote.
- *Trainable explicitly dependent methods*: In this group are the methods that use weights defined by the degree of satisfaction of the example to classify the different leaves reached and weights learned for the trees of the FRF ensemble.
- *Trainable implicity dependent methods*: In this group there are methods that use weights learned for the trees of the FRF ensemble.

We have presented experimental results obtained by applying the FRF ensemble to various datasets. Overall, the combination methods that achieve better performances are the weighted combination methods, compared to non-weighted methods typically used in random forest based ensembles. Among the weighted methods those using a weighting based on membership function have better performance, obtaining the best results in 65% of total tests performed. Although most of the methods of combination have the same computational cost, we highlight the increased cost of local fusion based methods. Nevertheless, these last methods have a good performance in the datasets with noise in the class attribute.

In particular, on the imperfect datasets (with missing and fuzzy values) the results obtained by the FRF ensemble are very promising. The FRF ensemble has a good performance with datasets with fuzzy values. The weighted combination methods perform better than non-weighted methods when working with these datasets.

With datasets with outliers the FRF ensemble shows a good performance and we can conclude that introducing outliers further away from the sample causes the FRF ensemble to behave just as if it had no outliers. When making the comparison with datasets with noise in the class attribute, the FRF ensemble shows a clear advantage over other proposals and the MWL-FUS2 combination method shows the best behaviour in most cases. So, the FRF ensemble is robust to noise.

When we compare the FRF ensemble with the base classifier, RF and ensembles using the same base classifier, the FRF ensemble obtains the best results. On comparing the results of the FRF ensemble with those obtained by a series of classifiers and multi-classifiers we conclude that the FRF ensemble is an effective classifier and that it obtains the best results in most cases.

Moreover, all these conclusions have been validated by applying statistical techniques to analyze the behaviour of different methods or algorithms compared in each experiment.

Acknowledgements

Supported by the project TIN2008-06872-C04-03 of the MICINN of Spain and European Fund for Regional Development. Thanks also to the Funding Program for Research Groups of Excellence with code 04552/GERM/06 granted by the “Fundación Séneca”, Murcia, Spain.

References

[1] H. Ahn, H. Moon, J. Fazzari, N. Lim, J. Chen, R. Kodell, Classification by ensembles from random partitions of high dimensional data, Computational Statistics and Data Analysis 51 (2007) 6166–6179.
[2] A. Asuncion, D.J. Newman, UCI Machine Learning Repository, University of California, School of Information and Computer Science, Irvine, CA, <http://www.ics.uci.edu/mlearn/MLRepository.html>.
[3] R.E. Banfield, L.O. Hall, K.W. Bowyer, W.P. Kegelmeyer, A comparison of decision tree ensemble creation techniques, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (1) (2007) 173–180.
[4] Y. Benjamini, Y. Hochberg, Controlling the false discovery rate: a practical and powerful approach to multiple testing, Journal of the Royal Statistical Society Series B 57 (1995) 289–300.
[5] P. Bonissone, F. Xue, R. Subbu, Fast meta-models for local fusion of multiple predictive models, Applied Soft Computing Journal (2008), doi:10.1016/j.asoc.2008.03.06.
[6] P.P. Bonissone, J.M. Cadenas, M.C. Garrido, R.A. Díaz-Valladares, A fuzzy random forest: fundamental for design and construction, in: Information Processing and Management of Uncertainty in Knowledge-Based Systems International Conference (IPMU2008), Málaga, Spain, 2008, pp. 1231–1238.
[7] L. Breiman, Bagging predictors, Machine Learning 24 (2) (1996) 123–140.
[8] L. Breiman, Random forests, Machine Learning 45 (1) (2001) 5–32.
[9] J. Casillas, L. Sánchez, Knowledge extraction from data fuzzy for estimating consumer behavior models, in: Proceeding Fuzzy IEEE, Vancouver, BC, Canada, 2006.
[10] J.M. Cadenas, M.C. Garrido, R.A. Díaz-Valladares, Hacia el Diseño y Construcción de un fuzzy random forest, in: Proceedings in II Simposio sobre Lógica Fuzzy y Soft Computing, Zaragoza, Spain, pp. 41–48.
[11] J.M. Cadenas, M.C. Garrido, R. Martínez, Una estrategia de particionamiento fuzzy basada en combinación de algoritmos, in: Proceedings in XIII Conferencia de la Asociación Española para la Inteligencia Artificial, Sevilla, Spain, 2009, pp. 379–388.
[12] J.M. Cadenas, J.V. Carrillo, M.C. Garrido, R. Martínez, E. Muñoz, NIP 1.5 – A tool to handle imperfect information on datasets, Murcia University, <http://heurimind.inf.um.es/NIP/index.htm>, 2008.

[13] T.G. Dietterich, An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization, Machine Learning 40 (2) (2000) 139–157.
[14] A. Fernández, M.J. del Jesus, F. Herrera, Hierarchical fuzzy rule based classification systems with genetic rule selection for imbalanced data-sets, International Journal of Approximate Reasoning 50 (3) (2009) 561–577.
[15] Y. Freud, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, Journal of Computer and System Sciences 55 (1) (1997) 119–139.
[16] S. García, A. Fernández, J. Luengo, F. Herrera, A study statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability, Soft Computing, doi:10.1007/s00500-0080392-y.
[17] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The WEKA data mining software: an update, SIGKDD Explorations 11 (1) (2009) 10–18.
[18] M. Hamza, D. Larocque, An empirical comparison of ensemble methods based on classification trees, Statistical Computation and Simulation 75 (8) (2005) 629–643.
[19] T.K. Ho, The random subspace method for constructing decision forests, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (8) (1998) 832–844.
[20] Y-C. Hu, A novel fuzzy classifier with Choquet integral-based grey relational analysis for pattern classification problems, Soft Computing 12 (2008) 523–533.
[21] J. Jang, Structure determination in fuzzy modeling: a fuzzy CART approach, in: Proceedings IEEE Conference on Fuzzy Systems, Orlando, USA, 1994, pp. 480–485.
[22] C.Z. Janikow, Fuzzy decision trees: issues and methods, IEEE Transaction on Systems, Man and Cybernetics, Part B 28 (1) (1998) 1–15.
[23] L. Koen-Myung, L. Kyung-Mi, L. Jee-Hyong, L. Hyung, A fuzzy decision tree induction method for fuzzy data, in: Proceedings IEEE Conference on Fuzzy Systems, Seoul, Korea, 1999, pp. 22–25.
[24] L.I. Kuncheva, Fuzzy vs Non-fuzzy in combining classifiers designed by boosting, IEEE Transaction on Fuzzy Systems 11 (6) (2003) 729–741.
[25] L.I. Kuncheva, Combining Pattern Classifiers: Methods and Algorithms, John Wiley & Sons, Inc., 2004.
[26] L.F. Mendonça, S.M. Vieira, J.M.C. Sousa, Decision tree search methods in fuzzy modeling and classification, International Journal of Approximate Reasoning 44 (2) (2007) 106–123.
[27] D. Opitz, R. Maclin, Popular Ensemble Methods: an empirical study, Journal of Artificial Intelligence 11 (1999) 169–198.
[28] A. Otero, J. Otero, L. Sánchez, J.R. Villar, Longest path estimation from inherently fuzzy data acquired with GPS using genetic algorithms, in: Proceedings on EFS, Lancaster, UK, 2006.
[29] P. Pulkkinen, H. Koivisto, Fuzzy classifier identification using decision tree and multiobjective evolutionary algorithms, International Journal of Approximate Reasoning 48 (2) (2008) 526–543.
[30] L. Sánchez, M.R. Suárez, J.R. Villar, I. Couso, Mutual information-based feature selection and partition design in fuzzy rule-based classifiers from vague data, International Journal of Approximate Reasoning 49 (3) (2008) 607–622.
[31] L. Sánchez, I. Couso, J. Casillas, Genetic learning of fuzzy rules based on low quality data, Fuzzy Sets and Systems 160 (2009) 2524–2552.
[32] R.E. Schapire, Theoretical views of boosting, in: Proceedings of 4th Eur. Conf. Computational Learning Theory, 1999, pp. 1–10.