Combination Methods in a Fuzzy Random Forest

P.P. Bonissone
GE Global Research
One Research Circle
Niskayuna, NY 12309. USA
bonissone@crd.ge.com

J.M. Cadenas M.C. Garrido
Dept. Ingeniería de la Información
y las Comunicaciones
Universidad de Murcia. Spain
jcadenas@um.es carmengarrido@um.es

R.A. Díaz-Valladares
Dept. Ciencias Computacionales
Universidad de Montemorelos
Mexico
rdiaz@um.edu.mx

***Abstract*—When individual classifiers are combined appropriately, we usually obtain a better performance in terms of classification precision. Multi-classifiers are the result of combining several individual classifiers. In this work we propose and compare various combination methods to obtain the final decision of the multi-classifier based on a "forest" of randomly generated fuzzy decision trees, i.e., a *Fuzzy Random Forest*. We propose various forms of weighting decisions on the basis of information obtained from the FRF. We make a comparative study with several databases to show the efficiency of the various combination methods.**

I. Introduction

Classification has always been a challenging problem, [1]. The explosion of information that today is available to companies and individuals further compounds this problem. We have witnessed a variety of methods and algorithms addressing the classification issue. In the last few years, we have also seen an increase of multi-classifiers based approaches, which have shown to deliver better results than individual classifiers, [18].

In [2] we propose a multi-classifier but we will not address the issue of how to obtain the best multi-classifier system. Rather, our focus will be on how to start from a multi-classifier system with performance comparable to the best classifiers and extend it to handle and manipulate imperfect information (linguistic labels, missing values, etc.)

To build the multi-classifier, we follow the methodology of random forest. To incorporate the processing of imperfect data, we construct the random forest using fuzzy trees as base classifiers. Therefore, we try to use the robustness of a tree ensemble, the power of the randomness to increase the diversity of the trees in the forest, and the flexibility of fuzzy logic and the fuzzy sets for data managing.

To combine the outputs of different classifiers, normally has been used as combination method the majority vote weighted method. However, this combination method is optimal in the case of two classes and classifiers with independent outputs. In addition, although the classifiers are designed independently, will be unlikely independent outputs [14]. Therefore, in this work we propose and compare various combination methods to obtain the final decision of the multi-classifier proposed in [2].

In section II, we review the major elements that constitute the multi-classifier and the combination methods that are frequently used to obtain the final decision in the multi-classifiers. In section III, we explain the learning and inference phases of the fuzzy random forest. In section IV, we define a series of combination methods for the fuzzy random forests, for which we presents some experimental results in V. Finally, we present our conclusions in section VI.

II. Multi-classifiers and combination methods

A. Multi-classifiers

When individual classifiers are combined appropriately, we usually obtain a better performance in terms of classification precision and/or speed to find a better solution. Multi-classifiers are the result of combining several individual classifiers. Multi-classifiers differ among themselves by their diverse characteristics: (1) the number and (2) the type of the individual classifiers; (3) the characteristics of the subsets used by every classifiers of the set; (4) the consideration of the decisions; and (5) the size and the nature of the training sets for the classifiers [14]

Segrera [18] divides the methods for building multi-classifiers in two groups: ensemble and hybrid methods. The first type, such as Bagging [3] and Boosting [17], induces models that merge classifiers with the same learning algorithm, while introducing modifications in the training data set. The second, type such as Stacking [20], creates new hybrid learning techniques from different base learning algorithms.

An ensemble uses the predictions of multiple base-classifiers, typically through majority vote or averaged prediction, to produce a final ensemble-based decision. The ensemble-based predictions typically have lower generalization error rates than the ones obtained by a single model. The difference depends on the type of base-classifiers used, ensemble size, and the diversity or correlation between classifiers [1]. Ahn [1] indicates that, over the last few years, three ensemble-voting approaches have received attention by researchers: boosting [17], bagging [3] and random subspaces [8].

1) Random Forest - A multi-classifier based on decision trees: Decision trees have been the basis for the most important works on multi-classifiers systems. As a result, the labeled of "forest" has been given to the set of trees working on the same classification problem.

In bagging, diversity is obtained by constructing each classifier with a different set of examples, which is obtained from the original training set by re-sampling with replacement. Bagging then combines the decisions of the classifiers using uniform-weighted voting. Bagging improves the performance of single classifiers by reducing the variance error. Breiman categorizes bagging decision trees as a particular instance of random forest classification techniques. A random forest is a tree-based ensemble that uses some kind of independent randomization in the construction of every individual classifier. Many variants of bagging and random forests with excellent classification performance have been developed in [16].

Hamza [7] concludes that: 1) Random Forests are significantly better than Bagging, Boosting and a single tree; 2) their error rate is smaller than the best one obtained by other methods; and 3) they are more robust to noise than the other methods. Consequently, random forest is a very good classification method with the following characteristics: (1) it's easy to use; (2) it does not require models, or parameters to select except for the number of predictors to choose at random at each node; and (3) it's relatively robust to noise.

2) Fuzzy Logic and Decision Trees: Decision tree techniques have proved to be interpretable, efficient and capable of treating with applications of great scale. However, they are highly unstable when small disturbances are introduced in data learning. Fuzzy logic offers an improvement in these aspects due to the elasticity of the fuzzy set's formalism. In previous work [11], [12], [13] we can find approaches in which fuzzy sets and their underlying approximate reasoning capabilities have been successfully combined with decision trees. This combination 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 immunity to noise, an extended applicability to uncertain or vague contexts, and a support for the comprehensibility of the tree structure, which remains the principal representation of the resultant knowledge.

B. Combination methods

In this section we provide a brief description of how the outputs of each classifier are combined to produce the final decision. The combination methods encountered in the literature can be include in one of the three following groups:

1) Abstract-level methods.
2) Rank-level methods.
3) Measure-level methods.

1) Abstract-level methods: In the methods of this group the output for each classifier is a label associated with the class and, a rule of combination merges the output of classifiers to provide the output of the multi-classifier. We can find two types of rules. The fixed rules, based on the simple majority vote, and the rules learned, as the based on the weighted majority vote and the based on the Bayes approach.

In the case of simple majority vote, each classifier has a vote with the same weight. The class with greater number of votes is that taken as final decision.

In the case of weighted majority vote, each classifier may have different weight in the final result [15]. One way to assign this weight is training each classifier individually and make an individually testing for data previously fixed.

The weight assigned to each classifiers vote is based on its performance in the test. If we assume that the classifiers are independent and that the results achieved in the test represent the effectiveness of classifier, is likely that this combination method works better than the simple majority vote. The class with greater weight is the final decision of the whole.

When the rules are based on the Bayes approach, the bayesian statistics is used to determine set optimal classifications. With an example of entry, is assigned the class with greater probability to posteriori. The rules seek to estimate, through a set of training/validation, these probabilities a posteriori. Two used forms are:

- Behavior-Knowledge Space [10]
- BeliefFunctions ([21]

This kind of combination rule is more complex and less used.

2) Rank-level methods: A classifier associates to each class a value or probability that indicates its degree of confidence in the classification of an example given. In this type of methods, this information is used to assign a range to each class. Each classifier provides a list ordered by range of classes. In this group, as in the previous, we can find methods with fixed rules and methods with trained rules. With fixed rules we have the Borda count method and, with rules learned the Borda count weighted method [19].

3) Measure-level methods: Some classifiers provide as output values of confidence for each class, given an example. These values of confidence can be interpreted as the probability of that example belongs to the various classes.

The combination methods in this group seeking, from the values of confidence of classifiers, a measure of confidence for each class. Within this group can find methods with fixed rules (simple average, product, maximum, minimum, etc.) and with learned rules (weighted average) [14]. In the simple averaging rule will be averaged outputs of individual classifiers. That is, add the probability of a same class in all classifiers and divide by the number of them. In the rule of product, are multiplying the probabilities of a same class in all classifiers.

With maximum and minimum rules seeks the maximum/minimum value respectively of the probabilities of one class in all classifiers. In all cases, once that final values is calculated for each class, the class with greater value calculated is predicted.

In the weighted rules from this group, introducing weight to the outputs of classifiers. One approach commonly used is introduce proportional weight to the accuracy of each individual classifier.

III. A Fuzzy Random Forest

The multi-classifier used in this work is a forest of fuzzy decision trees generated randomly (Fuzzy Random Forest), following Breiman's methodology [4]. In this section we describe the learning phase to construct these multi-classifiers and the inference phase.

A. Fuzzy Random Forest Learning

To generate a Fuzzy Random Forest (FRF) we use the Algorithm 1.

1) Dividing the examples set of entry in subsets according to: Take a random sample of N observations from the data set with replacement of the complete set of M observations. Some observations will be selected more than once, and others will not be chosen. Approximately $2/3$ of the observations will be selected. The remaining $1/3$ of the cases is called "out of bag" (OOB). For each constructed tree, a new random selection of cases is performed.
2) For each subset of examples, apply algorithm 2 (construct a fuzzy tree). This algorithm has been adapted so that the trees could be constructed without considering all the attributes to split the nodes. We select the set of attributes as a random subset of the total set of available attributes. Perform a new random selection for each split. Some attributes (inclusive the best) cannot be considered for each split, but a attribute excluded in one split may be used by other splits in the same tree.
3) Repeat steps 1 and 2 up to building all fuzzy trees. At the end, we will have constructed a fuzzy random forest.

Algorithm 1. FRF Learning

Each tree in the forest will be a fuzzy tree generated following the guidelines of [12], adapting it where is necessary.

1) Start with examples set of entry, having the weights of the examples (in the root node) equal to 1.
2) At any node N still to be expanded, compute the number of examples of each class. The examples are distributed in part or in whole by branches. The distributed amount of each example to a branch is obtained as the product of its current weight and the membership degree to the node.
3) Compute the standard information content.
4) At each node search the set of remaining attributes to split the node.
   - Select with any criteria, the candidate attributes set to split the node.
   - Compute the standard information content to each child node obtained from each candidate attribute.
   - Select the candidate attribute such that information gain is maximal.
5) Divide N in sub-nodes according to possible outputs of the attribute selected in the previous step.
6) Repeat steps 2-5 to stop criteria is satisfied in all nodes.

Algorithm 2. Fuzzy Decision Tree Learning

With the Algorithm 1, we integrate the concept of fuzzy tree within the design philosophy of Breiman's random forest. In this way, we augment the capacity of diversification of random forests with the capacity of approximate reasoning of fuzzy logic.

When we constructed a random forest using the previous algorithm, about $1/3$ of the cases are excluded from each tree in the forest. These cases are called the "out of bag" (OOB); each tree will have a different set of OOB cases. The OOB cases are not used to build the tree and constitute an independent test sample for the tree.

As we shall see later, we use the OOB cases to obtain a learned combination method.

B. Fuzzy Random Forest Inference

For inference, we suggest wide-interpretation techniques of the class from the context fuzzy random forest. Now, we introduce the notation that we will use:

- T is the trees' number of the forest.
- N_t is the number of reached leaf nodes by an example, in the tree t. A characteristic inherent in fuzzy trees is that the classification of an example can derive in two or more leaves due to the overlapping of the fuzzy sets.
- I is the number of classes that we consider.
- $\chi_{t,n}(e)$ is the grade which the example e active the leaf n from t tree.
- $\omega_{t,n}$ is a vector with I elements indicating the weight of the I possible classes in the leaf n of tree t, $\omega_{t,n}$=$(\omega_{t,n,1}, \omega_{t,n,2}, ..., \omega_{t,n,I})$, where $\omega_{t,n,i} = \frac{E_i}{\sum_{j=1}^{I} E_j}$ and E_j is the number of examples with class j in the leaf.

To obtain information about the classes that provides a leaf reached n in a tree we define the function $K(\cdot,\cdot,\cdot)$. This function depends of $\chi_{t,n}(e)$ and vector $\omega_{t,n}$ and returns the weight assigned by the node n to class i. Examples of this function K given a reached leaf n in tree t can be:
$K(i, \chi_{t,n}(e), \omega_{t,n})$ provides:

- weight 1 if i is the majority class in this node and 0 for all other classes. It will be denoted by K1.
- the weight $\omega_{t,n,i}$ if i is the majority class in this node and 0 for all other classes (K2).
- the weight $\chi_{t,n}(e)$ if i is the majority class in this node and 0 for all other classes (K3).
- the weight $\chi_{t,n}(e) \cdot \omega_{t,n,i}$ if i is the majority class in this node and 0 for all other classes (K4).
- the weight $\omega_{t,n,i}$ for each class i (K5).
- the weight $\chi_{t,n}(e) \cdot \omega_{t,n,i}$ for each class (K6).

The fuzzy classifier module operate on fuzzy trees of the forest with two possible strategies:

Strategy 1: Combining the information from the different reached leaves in each tree to obtain the decision of each individual tree and then apply the same one or another combination method to generate the global decision of the forest (Fig. 1).

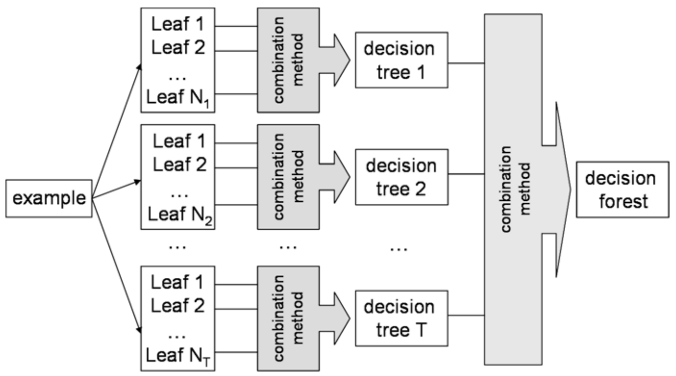

Fig. 1. Inference with strategy 1

Strategy 2: Combining the information from all reached leaves from all trees to generate the global decision of the forest (Fig. 2).

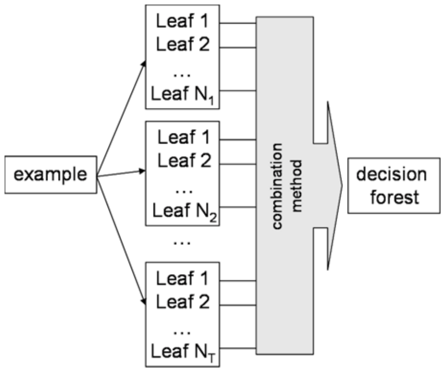

Fig. 2. Inference with strategy 2

In Algorithm 3, we use the function $Faggre$, which is defined as a frequently used multi-classifiers combination method [14], e.g., majority vote, minimum, maximum, average, product, etc. First, $Faggre$ is used to obtain $A[t][i]$ (the weight of each tree for each class). Later, the values obtained in each tree t, will be aggregating by means of the function $Faggre$ (again, can be any combination method mentioned previously, being able to adapt to take care about some considerations commented ahead) to obtain the vector F that contains the weight proposed by fuzzy random forest for the different classes.

```
Begin
  TreeClasification.
  ForestClasification.
End

TreeClasification ($Random\ Forest, A[t][i]$)
Begin
  For each Tree $t$
    For each Class $i$
      $A[t][i] = Faggre(K(1, \chi_{t,1}(e), \omega_{t,1}), ...,$
          $K(1, \chi_{t,N_t}(e), \omega_{t,N_t}), ..., K(I, \chi_{t,1}(e), \omega_{t,1}),$
          $, ..., K(I, \chi_{t,N_t}(e), \omega_{t,N_t}))$
    End For each Class
  End For each Tree
End

ForestClasification ($A[t][i]$, $F[i]$)
Begin
  For each Class $i$
    $F[i] = Faggre(A[1][i], ..., A[T][i])$
  End For each Class
End
```

Algorithm 3. FRF Inference (Strategy 1)

For implementing strategy 2, the previous algorithm is simplified so that it does not add the information for tree, but provides directly the information of all leaves reached by the example e in the different trees of the forest.

```
ForestClasification ($Random\ Forest, F[i]$)
Begin
  For each Class $i$
    $F[i] = Faggre(K(i, \chi_{1,1}(e), \omega_{1,1}), ...,$
        $K(i, \chi_{1,N_1}(e), \omega_{1,N_1}), ..., K(i, \chi_{T,1}(e), \omega_{T,1}),$
        $..., K(i, \chi_{T,N_T}(e), \omega_{T,N_T}))$
  End For each Class
End
```

Algorithm 4. FRF Inference (Strategy 2)

IV. Defining Combination Methods

In the previous section we have shown the general scheme of inference that we use to get the final decision of the forest. In this section, we describe specific instances of inference that we have implemented for both strategies. These specific instances use the combination methods commented in previous sections to implement $Faggre$ function. In the next section show some experimental results.

We have divided the several implementations in three groups depending on the combination method used in each of them to implement $Faggre$: majority vote, minimum and maximum. As in the implementations of the strategy 1, two combination methods are used, we only group them in one of the previous groups.

• **Majority vote -** Within this group we define the following implementations:

○ *Simple Majority Vote applied to strategy 1 (SM1)*: The forest assigns to e the class c if:

$$c = max_{i=1}^{I} \sum_{t=1}^{T} A[t][i]$$

where $A[t][i]$ is obtained using the same combination method but applied to the values of reached leaves in each tree. Each reached leaf provides values to each class using the $K1$ function:

$$A[t][i] = \begin{cases} 1 & \text{if } i = \max_{j=1}^{I} \sum_{n=1}^{N_t} K(j, \chi_{t,n}(e), \omega_{t,n}) \\ 0 & \text{other case} \end{cases}$$

○ *Simple Majority Vote applied to strategy 2 (SM2)*: Every reached leaf of the forest is considered to vote. The forest will decide the class c if

$$c = \max_{i=1}^{I} \sum_{t=1}^{T} \sum_{n=1}^{N_t} K(i, \chi_{t,n}(e), \omega_{t,n})$$

Again, each reached leaf provides values to each class using the $K1$ function.

Weighted versions of these strategies are the following:

○ *Majority Vote Weighted by Leaf applied to strategy 1 (MWL1)*: is defined as SM1 but uses the $K3$ function to introduce the weight of the leaf in the decision of each tree.

○ *Majority Vote Weighted by Leaf applied to strategy 2 (MWL2)*: is defined as SM2 but uses the $K3$ function to introduce the weight of the leaf in the decision of each tree.

○ *Majority Vote Weighted by Leaf and by Tree applied to strategy 1 (MWLT1)*: Is defined as SM1 but uses the $K3$ function and a weight for each tree obtained by testing each individual tree with the OOB data file commented in the previous section. Lets be $p = (p_1, p_2, ..., p_T)$ the vector with the weights assigned to each tree. Each p_i is obtained as:

$$\frac{N\_success\_OOB_i}{size\_OOB_i}$$

where $N\_success\_OOB_i$ is the number of examples inferred correctly from the OOB file used for testing the i th tree and $size\_OOB_i$ is the total number of examples in this file.

Once the vector p is obtained, we used it in the final decision of forest, so that the forest votes for the class c if:

$$c = \max_{i=1}^{I} \sum_{t=1}^{T} p_t \cdot A[t][i]$$

○ *Majority Vote Weighted by Leaf and by Tree (MWLT2)*: is defined as SM2 but uses the $K3$ function and the vector of weights p applied to strategy 2. In this case, the forest votes for the class c if

$$c = \max_{i=1}^{I} \sum_{t=1}^{T} p_t \sum_{n=1}^{N_t} K(i, \chi_{t,n}(e), \omega_{t,n})$$

• **Minimum -** Within this group we define the following implementations:

○ *Minimum Simple applied to strategy 1 (MIS1)*: In strategy 1, we have applied the combination method minimum at decision level of every tree, in which case for each class is selected the minimum value of the all reached leaves in the tree. For this we use the $K5$ function. The tree votes for the class with greater value:

$$A[t][i] = \begin{cases} 1 & \text{if } i = \max_{j=1}^{I} min(K(j, \chi_{t,1}(e), \omega_{t,1}), ... \\ & ..., K(j, \chi_{t,N_t}(e), \omega_{t,N_t})) \\ 0 & \text{other case} \end{cases}$$

The forest uses the simple vote majority method. It votes for the class c if:

$$c = max_{i=1}^{I} F[i] = \sum_{t=1}^{T} A[t][i]$$

○ *Minimum Simple applied to strategy 2 (MIS2)*: If we apply the minimum method at decision level of the forest, we obtain the values

$$F[i] = min(K(i, \chi_{1,1}(e), \omega_{1,1}), ..., K(i, \chi_{1,N_1}(e), \omega_{1,N_1}), ..$$
$$.., K(i, \chi_{T,1}(e), \omega_{T,1}), ..., K(i, \chi_{T,N_T}(e), \omega_{T,N_T}))$$

where we use the $K5$ function.

Then, the forest votes for the class c with the most high value as label suggested by classifier:

$$c = max_{i=1}^{I} F[i]$$

• **Maximum -** Within this group we have defined implementations equivalent to the minimum group but applying the maximum operator at all sites where the minimum operator is applied. Thus we define:

○ *Maximum Simple applied to strategy 1 (MAS1).*

○ *Maximum Simple applied to strategy 2 (MAS2).*

V. Experiments and preliminary results

To illustrate the behavior of different implementations, we have selected four databases used as benchmarks in the recent literature: Iris, Diabetes, Ionosphere and Wine databases.

For each one, we have generated 5 versions of the original database which include various types of imperfect information because, as we commented earlier, the principal aim we seek with the FRF multi-classifier is to obtain similar results to the most classifiers and in addition to work with imperfect data. In this way, we have generated databases with a 10% of linguistic labels (each of the domains of the numeric attributes is partitioned into a set of linguistic labels and 10% of the numerical values in the database are replaced by their labels), a 15% of linguistic labels, a 10% of missing values and a 15% of missing values. Forests have been generated with random initial parameters for each database. Also we have executed the technique FID [12] that uses a single tree generated by the algorithm 2 with random initial parameters. We are interested in this technique because allows the management of imperfect information, in particular of linguistic labels.

In tables I, II, III and IV we show the obtained results of ten-fold cross-validation, showing the accuracy average and the average standard deviation. As we can see, implementations based on the majority vote have the best behavior and it is interesting include theirs weighted versions, because obtain good results.

The implementation based on the minimum applied to strategy 2 obtains poor results. This is because there are a large number of reached leaves and this produces that this method often assigns 0 to each class, because of this the forest is unable to take a decision. Finally, implementations based on the maximum work fairly well with both strategies.

With regard to the comparative results of the management of imperfection made by both techniques (FID and FRF), we can see that the multi-classifier FRF improvement in almost all cases the performance of FID.

VI. Summary

In this paper we have defined various implementations to combine the outputs of classifiers that composing the multi-classifier FRF. These implementations are based on the combination methods used frequently in the literature to get the final decision in multi-classifiers. In this way we have defined:

- Implementations based on majority vote method.
- Implementations based on minimum method.
- Implementations based en el maximum method.

We have presented experimental results obtained by applying these implementations to various databases. Overall, the several implementations have a good performance except based on the minimum method applied to strategy 1.

TABLE I
TESTING ACCURACIES FOR THE COMBINATION METHODS ON IRIS DATABASE

Comb.	without imp.	10% labels	15% labels	10% missing	15% missing
FID	97.33 (4.66)	94.67 (6.13)	96.00 (4.66)	84.67 (7.73)	86.67 (11,76)
MAS1	97.33 (4.66)	96.00 (5.62)	95.33 (4.50)	91.86 (7.83)	93.08 (5.75)
MAS2	97.33 (4.66)	95.29 (5.51)	94.67 (5.26)	95.23 (4.56)	92.67 (7.98)
SM1	97.33 (4.66)	96.62 (4.75)	96.67 (3.52)	88.67 (7.73)	86.52 (12.27)
MWL1	97.33 (4.66)	96.00 (5.62)	95.33 (4.50)	89.33 (7.82)	87.19 (11.64)
MWLT1	97.33 (4.66)	96.00 (5.62)	95.33 (4.50)	89.33 (7.82)	86.67 (12.17)
SM2	97.33 (4.66)	96.62 (4.75)	96.67 (3.52)	88.67 (7.73)	86.00 (12.75)
MWL2	97.33 (4.66)	96.00 (5.62)	95.33 (4.50)	92.67 (7.34)	86.67 (12.17)
MWLT2	97.33 (4.66)	96.00 (5.62)	95.33 (4.50)	92.67 (7.34)	86.67 (12.17)
MIS1	97.29 (4.71)	95.90 (5.80)	96.62 (3.57)	93.95 (5.84)	93.81 (7.46)
MIS2	89.33 (7.82)	88.67 (8.92)	88.67 (7.06)	78.00 (5.49)	76.00 (9.00)

TABLE II
TESTING ACCURACIES FOR THE COMBINATION METHODS ON DIABETES DATABASE

Comb.	without imp.	10% labels	15% labels	10% missing	15% missing
FID	73.69 (4.54)	70.44 (8.41)	66.40 (6.40)	73.95 (4.56)	69.13 (6.81)
MAS1	73.56 (3.72)	74.09 (6.02)	73.16 (7.82)	73.30 (4.25)	72.39 (4.91)
MAS2	72.78 (4.86)	73.70 (6.55)	74.73 (5.83)	73.96 (4.76)	72.00 (6.39)
SM1	74.73 (3.65)	73.70 (6.47)	73.17 (7.72)	73.69 (3.95)	74.21 (4.51)
MWL1	73.82 (3.57)	73.96 (6.34)	73.03 (8.03)	73.82 (4.07)	73.39 (4.19)
MWLT1	73.82 (3.57)	73.96 (6.34)	74.33 (6.83)	73.95 (4.05)	73.05 (4.76)
SM2	74.34 (3.36)	73.96 (5.80)	73.17 (7.72)	73.69 (3.28)	73.26 (4.00)
MWL2	73.82 (3.57)	73.57 (5.87)	73.69 (7.40)	73.95 (4.01)	72.65 (4.84)
MWLT2	73.82 (3.57)	73.57 (5.87)	74.47 (6.59)	73.95 (4.01)	73.53 (3.68)
MIS1	74.34 (3.60)	74.22 (6.20)	73.69 (8.31)	73.82 (3.84)	71.44 (4.78)
MIS2	68.36 (3.80)	72.66 (5.04)	71.60 (5.99)	70.18 (3.81)	68.96 (4.33)

TABLE III
TESTING ACCURACIES FOR THE COMBINATION METHODS ON IONOSPHERE DATABASE

Comb.	without imp.	10% labels	15% labels	10% missing	15% missing
FID	90.01 (4.11)	92.32 (4.02)	92.57 (4.51)	85.48 (4.49)	84.90 (7.27)
MAS1	92.58 (5.08)	92.59 (3.81)	93.40 (4.10)	90.60 (5.73)	85.74 (5.02)
MAS2	94.87 (4.57)	94.25 (3.82)	94.26 (2.15)	91.62 (3.45)	87.08 (5.93)
SM1	92.29 (5.40)	91.97 (4.25)	93.68 (4.26)	90.77 (6.29)	87.73 (5.57)
MWL1	92.58 (5.08)	92.59 (3.81)	93.11 (4.13)	91.10 (4.83)	86.24 (4.25)
MWLT1	92.58 (5.08)	92.59 (3.81)	93.14 (4.09)	90.60 (5.38)	86.60 (4.69)
SM2	92.56 (4.93)	92.30 (3.78)	93.43 (4.05)	91.92 (5.94)	87.67 (5.25)
MWL2	92.30 (5.40)	92.32 (3.79)	93.68 (4.26)	91.74 (4.92)	88.32 (4.56)
MWLT2	92.30 (5.40)	92.32 (3.79)	93.71 (4.22)	91.74 (4.92)	87.18 (4.33)
MIS1	91.98 (5.02)	91.75 (4.92)	93.68 (4.26)	90.52 (7.68)	85.95 (4.98)
MIS2	80.32 (7.83)	83.78 (4.38)	85.46 (6.40)	69.52 (11.02)	69.52 (9.71)

TABLE IV
TESTING ACCURACIES FOR THE COMBINATION METHODS ON WINE DATABASE

Comb.	without imp.	10% labels	15% labels	10% missing	15% missing
FID	94.41 (4.54)	93.86 (6.65)	94.90 (4.22)	80.98 (7.35)	90.42 (5.40)
MAS1	96.08 (5.90)	94.38 (3.71)	94.90 (4.22)	90.98 (7.94)	93.30 (5.10)
MAS2	94.93 (5.53)	94.93 (3.17)	94.24 (3.96)	86.96 (10.24)	89.38 (6.60)
SM1	94.97 (5.53)	93.24 (5.79)	94.93 (4.87)	90.04 (7.91)	93.30 (3.50)
MWL1	96.08 (5.90)	93.82 (4.86)	94.90 (4.22)	90.15 (7.58)	96.11 (4.57)
MWLT1	96.08 (5.90)	93.82 (4.86)	94.90 (4.22)	90.42 (6.99)	95.52 (4.39)
SM2	94.97 (5.53)	93.24 (5.79)	94.93 (4.87)	90.71 (6.96)	93.76 (3.13)
MWL2	96.08 (5.90)	94.38 (3.71)	94.90 (4.22)	91.46 (6.12)	94.93 (4.11)
MWLT2	96.08 (5.90)	94.38 (3.71)	94.90 (4.22)	90.95 (6.68)	94.93 (4.11)
MS1	95.49 (4.40)	93.82 (6.18)	93.79 (6.65)	83.73 (4.82)	93.82 (4.86)
MS2	84.31 (6.74)	91.01 (4.70)	87.55 (7.12)	66.37 (10.15)	73.56 (11.95)

The results obtained with multi-classifier FRF have been compared with those obtained with FID, with the aim of analyze the management of imperfect information on both techniques. The obtained results are quite promising to multi-classifier FRF.

ACKNOWLEDGMENT

The authors thank the “Ministerio de Educación y Ciencia” of Spain and the European Fund of Regional Development for the support given to this work under the project TIN2005-08404-C04-02 and the “Fundación Séneca, Agencia de Ciencia y Tecnología de la Región de Murcia (Plan Regional de Ciencia y Tecnología 2007/2010)” which give support under the “Programa de Ayudas a Grupos de Excelencia de la Región de Murcia”.

REFERENCES

[1] H. Ahn, H. Moon, J. Fazzari, N. Lim, J. Chen, R. Kodell (2007). Classification by ensembles from random partitions of high dimensional data. *Computational Statistics and Data Analysis* 51, 6166–6179.

[2] P.P. Bonissone, J.M. Cadenas, M.C. Garrido, R.A. Díaz-Valladares (2008). A fuzzy Random Forest: Fundamental for Design and Construction. *Information Processing and Management of Uncertainty in Knowledge-Based systems International Conference (IPMU2008)* Málaga, Spain, Accept and publication pending.

[3] L. Breiman (1996). Bagging predictors. *Machine Learning* 24(2), 123–140.

[4] L. Breiman (2001). Random forests. *Machine Learning* 45(1), 5–32.

[5] J.M. Cadenas, M.C. Garrido, R.A. Díaz-Valladares. Hacia el Diseño y Construcción de un Fuzzy Random Forest. *Proc. in II Simposio sobre Lógica Fuzzy y Soft Computing*, 41–48, Zaragoza, Spain.

[6] T.G. Dietterich (2000). An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization. *Machine Learning* 40(2), 139–157.

[7] M. Hamza, D. Larocque (2005). An empirical comparison of ensemble methods based on classification trees. *Statistical Computati. & Simulation* 75(8), 629-643.

[8] T.K. Ho (1998). The random subspace method for constructing decision forests. *Transactions on Pattern Analysis and Machine Intelligence* 20(8), 832–844.

[9] T. Hothorn, B. Lausen (2003). Double-bagging: combining classifiers by bootstrap aggregation. *Pattern Recognition* 36(6), 1303–1309.

[10] Y.S. Huang, C.Y. Suen (1995). A method for combining multiple experts for the recognition of unconstrained handwritten numerals *IEEE Transactions Pattern Analysis and Machine Intelligence* 17, 90–93.

[11] J. Jang (1994). Structure determination in fuzzy modeling: A Fuzzy CART approach. *Proceedings IEEE Conference on Fuzzy Systems*, 480–485, Orlando, USA.

[12] C.Z. Janikow (1998). Fuzzy decision trees: issues and methods. *Transaction on Systems, Man and Cybernetics, Part B* 28(1), 1–15.

[13] L. Koen-Myung, L. Kyung-Mi, L. Jee-Hyong, L. Hyung (1999). A Fuzzy Decision Tree Induction Method for Fuzzy Data, *Proceedings IEEE Conference on Fuzzy Systems*, 22–25, Seoul, Korea.

[14] L.I. Kuncheva (2003). Fuzzy vs Non-fuzzy in combining classifiers designed by boosting. *IEEE Trans. on Fuzzy Systems* 11(6), 729–741.

[15] N. Littlestone, M.K. Warmuth (1994). The weighted majority algorithm *Information and Computation* 108, 212–261.

[16] G. Martínez-Muñoz, A. Suárez (2005). Switching class labels to generate classification ensembles. *Pattern Recognition* 38(10), 1483–1494.

[17] R.E. Schapire (1990). The strength of weak learnability. *Machine Learning* 5(2), 197-227.

[18] S. Segrera, M. Moreno (2006). An Experimental Comparative Study of Web Mining Methods for Recommender Systems. *Proc. of the 6th WSEAS Intern. Conference on Distance Learning and Web Engineering*, 56–61, Lisbon, Portugal.

[19] M. Van Erp, L. Schomaker (2000). Variants of the Borda count method for combining ranked classifier hypotheses. *Proceedings of the Seventh International Workshop on Frontiers in Handwriting Recognition* Amsterdam, September, 443–452.

[20] D. Wolpert (1992). Stacked Generalization. *Neural Networks* 5, 241–259.

[21] L. Xu, A. Krzyzak, C.Y. Suen (1992). Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man and Cibernetics* 22(3), 418–435.