Weighted decisions in a Fuzzy Random Forest

P.P. Bonissone[1] J.M. Cadenas[2] M.C. Garrido[2] R.A. Díaz-Valladares[3] R. Martínez[2]

1. GE Global Research. One Research Circle.
Niskayuna, NY 12309. USA
2. Dept. Ingeniería de la Información y las Comunicaciones. Universidad de Murcia.
Murcia, Spain
3. Dept. Ciencias Computacionales. Universidad de Montemorelos.
Montemorelos, México
Email: bonissone@crd.ge.com,{jcadenas,carmengarrido}@um.es,rdiaz@um.edu.mx,raquel.m.e@alu.um.es

Abstract— *A multi-classifier system - obtained by combining several individual classifiers - usually exhibits a better performance (precision) than any of the original classifiers. In this work we use a multi-classifier based on a forest of randomly generated fuzzy decision trees (Fuzzy Random Forest), and we propose a new method to combine their decisions to obtain the final decision of the forest. The proposed combination is a weighted method based on the concept of local fusion and on the data set Out Of Bag (OOB) error.*

Keywords— combination methods, fuzzy trees, local fusion, multi-classifier, random forest

1 Introduction

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 [10].

The multi-classifier Fuzzy Random Forest [2], doesn't aim to obtain the best system multi-classifier. Rather, we will focus 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, noise in nominal attributes, etc.) maintaining a proper performance. To build the multi-classifier, we follow the methodology of random forest and 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 fuzzy sets for imperfect data managing.

Usually, the majority vote weighted method has been the most common combination method used to combine the outputs of different classifiers. This combination method is optimal in the case of two classes and classifiers with independent outputs. In addition, although the classifiers are designed independently, it will be unlikely that they produce independent outputs [9]. In [2] we have presented various combination methods to obtain the final decision of the proposed multi-classifier.

In this work, we propose new inference implementations to obtain the decision of the multi-classifier Fuzzy Random Forest and we compare them with two of the best performing implementations in Fuzzy Random Forest: the proposed implementations are weighted implementations based on local fusion concept [3] and we compare them with weighted implementations based in the OOB cases.

In the section 2, we review the major elements that constitute a multi-classifier and some attempts for the incorporation of fuzzy logic in some techniques. In section 3, we provide a brief description of the Fuzzy Random Forest multi-classifier, showing the learning and inference phases. In section 4, we present the inference implementations that are the subject of this work. We present some comparative results of the analyzed implementations in section 5. Finally, we present our conclusions in section 6.

2 Multi-classifiers and Fuzzy Logic

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: the number and the type of the individual classifiers; the characteristics of the subsets used by every classifiers of the set; the consideration of the decisions; and the size and the nature of the training sets for the classifiers [9].

Segrera [10] divides the methods for building multi-classifiers in two groups: ensemble and hybrid methods. The first type, such as Bagging and Boosting, induces models that merge classifiers with the same learning algorithm, while introducing modifications in the training data set. The second, type such as Stacking, 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, bagging and random subspaces.

2.1 *Random Forest: A Multi-Classifier based on Decision Trees*

Breiman [4] further defines a random forest as a classifiers composed by decision trees where every tree h_t has been generated from the set of data training and a vector θ_t of random numbers identically distributed and independent from the vectors θ_1, θ_2,..,θ_{t-1} used to generate the classifiers $h_1, h_2, .., h_{t-1}$. Each tree provides his unitary vote for the majority class given the entry. Examples of random forest are: randomization, Forest-RI and Forest-RC, double-bagging.

Hamza [6] concludes that: Random Forests are significantly better than Bagging, Boosting and a single tree; their error rate is smaller than the best one obtained by other methods; and they are more robust to noise than the other methods. Consequently, random forest is a very good classification method with the following characteristics: it's easy to use; it does not require models, or parameters to select except for the number of predictors to choose at random at each node.

2.2 *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. For this reason, the fuzzy logic has been incorporated in the decision tree construction techniques.

Fuzzy logic offers an improvement in the disadvantages previously commented on the decision trees due to the elasticity of the fuzzy set's formalism. In [7, 8] 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.

In the literature, we can find several proposals for building trees of fuzzy information starting with algorithms already known for building traditional trees. Fuzzy CART [7] was one of the first examples of this approach, being based on the CART algorithm. However, most authors have preferred to use the ID3 algorithm to construct trees for recursive partition of the data set of agreement to the values of the selected attribute. To use the ID3 algorithm in the construction of fuzzy trees, we need to develop attribute value space partitioning methods, branching attribute selection method, branching test method to determine the degree to which data follow the branches of a node, and leaf node labeling methods to determine classes [8].

Fuzzy decision trees are constructed in a top-down manner by recursive partitioning the training set into subsets. Some particular features of fuzzy tree learning are: the member-ship degree of examples, the selection of test attributes, the fuzzy tests (to determine the membership degree of the value of an attribute to a fuzzy set), and the stop criteria (besides the classic criteria when the measure of the information is under a specific threshold) [8].

3 Fuzzy Random Forest

Following Breiman's methodology [4], FRF is a multi-classifier that is a forest of randomly-generated fuzzy decision trees. In this section we provide a brief description of the learning and inference phases in FRF.

3.1 *Fuzzy Random Forest Learning*

To built a Fuzzy Random Forest (FRF) we use the algorithm 1.

```
FRFLearning (M,Fuzzy Random Forest)
begin
  1. Take a random sample of $N$ observations from the data set
  with replacement of the complete set of $M$ observations.
  2. Apply algorithm 2 (construct a fuzzy tree) to the subset of
  examples.
  3. Repeat steps 1 and 2 up to building all fuzzy trees.
end.
```

Algorithm 1. FRF learning

Each tree in the forest will be a fuzzy tree generated following the guidelines of [8], adapting it where is necessary. So, the following algorithm (Algorithm 2) has been designed 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. Then, we perform a new random selection for each split. As in the original case, some attributes (inclusive the best) might not be considered for each split, but a attribute excluded in one split might be used by other splits in the same tree.

```
FuzzyDecisionTree(Examples,Fuzzy Tree)
begin
  1. Start with examples set of entry, having the weights of the
  examples (in 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.
    4.1.Select with any criteria, the candidate attributes set to
    split the node.
    4.2. Compute the standard information content to each child
    node obtained from from each candidate attribute.
    4.3.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.
end.
```

Algorithm 2. Fuzzy decision tree learning

With Algorithm 2, 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 built a Fuzzy Random Forest using the previous algorithm, about $1/3$ of the cases are excluded from each tree

ISBN: 978-989-95079-6-8

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. We use the set of OOB cases to obtain one of the weighted combination methods proposed in this work.

3.2 *Fuzzy Random Forest Inference*

In this section we will describe how the inference is carried out using Fuzzy Random Forest. First, we introduce the notation that we will use in the following sections. Then, we define two general strategies to obtain the decision of the forest for a target example. Concrete instances of these strategies are defined in the next section where we present different inference implementations in the Fuzzy Random Forest.
Notations

- T is the trees’ number of the forest. We will use the index t to refer to a particular tree in 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. We will use the index n to refer to a particular leaf in a tree of the forest.
- I is the number of classes that we consider. We will use the index i to refer to a particular class.
- e is an example from the dataset used to built and to infer with the Fuzzy Random Forest.
- $\chi_{t,n}(e)$ is the grade which the example e active the leaf n from t tree.
- $\overline{\omega}_{t,n}$ is a vector with I elements indicating the weights of the I possible classes in the leaf n of tree t, $\overline{\omega}_{t,n}$=$(\omega_{t,n,1}, \omega_{t,n,2}, ..., \omega_{t,n,I})$, where $\omega_{t,n,i} = \frac{E_i}{\sum_{j=1}^{I} E_j}$ and E_j is the sum of the weights of examples with class j in the leaf.

To make the inference with the Fuzzy Random Forest, we will define the fuzzy classifier module. 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).

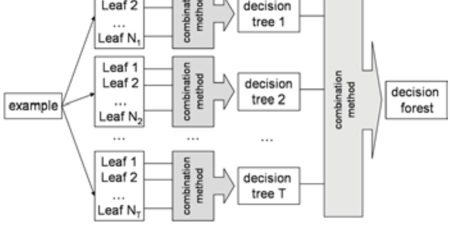

Figure 1: Inference with strategy 1

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

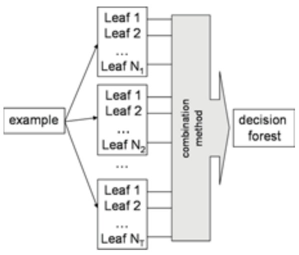

Figure 2: Inference with strategy 2

Moreover, in strategies 1 and 2 we use two functions: $K(\cdot, \cdot, \cdot)$ and $Faggre$. With $K(\cdot, \cdot, \cdot)$ we obtain information about the classes that provides a leaf reached n in a tree. This function depends of $\chi_{t,n}(e)$ and vector $\overline{\omega}_{t,n}$ and returns the weight assigned by the node n to class i. Concrete cases of this function will be defined in the next section. And, $Faggre$ is defined as a frequently used multi-classifiers combination method [9], e.g., majority vote, minimum, maximum, average, product or a weighting combination method defined in the next section.

In Algorithm 3 we implement strategy 1. In this algorithm first, $Faggre$ is used to obtain $A[t][i]$ (the weight of each tree for each class). In this case, $Faggre$ aggregates the information provided by the reached leaves in a tree. Later, the values obtained in each tree t, will be aggregating by means of the function $Faggre$ to obtain the vector $\overline{F}$ that contains the weight proposed by the Fuzzy Random Forest for the different classes. This algorithm takes a target example e and the random forest, and generates the vector $\overline{F}$ as output.

```
Begin
  TreeClasification.
  ForestClasification.
End

TreeClasification ($in : e, in : Random\ Forest, out : A$)
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 ($in : A, out : \overline{F}$)
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)

To implement strategy 2, the previous algorithm is simplified so that it does not add the information for each tree, but

ISBN: 978-989-95079-6-8

uses directly the information of all leaves reached by the example e in the different trees of the forest. The algorithm 4 implement the strategy 2 and uses as target values the example e to classify and the random forest and provides the vector $\overline{F}$ that contains the weight proposed by fuzzy random forest for the different classes.

```
ForestClasification ($in : e, in : Random\ Forest, out : \overline{F}$)
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)

4 Inference Implementations

In the previous section we have shown the general scheme of inference that we use to obtain the forest's final decision. In [2] specific inference instances to FRF are described to both strategies. These specific instances use the combination methods commented in previous sections to implement $Faggre$ function. To implement concrete instances of the strategies we need to define particular examples to the function K previously defined.

Given a reached leaf n in tree t, the particular case of $K(i, \chi_{t,n}(e), \omega_{t,n})$ that we use provides the weight $\chi_{t,n}(e)$ if i is the majority class in this node and 0 for all other classes. This particular case is denoted K1.

In this work we present new inference implementations to FRF based on local fusion concept. Also, we compare the results of these new implementations with the results obtained to the weighted implementations based on the OOB cases. Next we describe these implementations:

• **Weighted Implementations using the OOB cases-** Within this group we define the following implementations:

- *Majority Vote Weighted by Leaf and by Tree applied to strategy 1* (MWLT1): Uses a weight for each tree obtained by testing each individual tree with the OOB data file commented in the previous section. Lets be $\overline{p} = (p_1, p_2, ..., p_T)$ 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 $\overline{p}$ is obtained, we used it in the final decision of forest, so that the forest votes for the class c if:

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

  where $A[t][i]$ is obtained using the majority vote 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:

  $$A[t][i] = \begin{cases} 1 & \text{if } i = \max_{j=1}^{I} \sum_{n=1}^{N_t} K1(j, \chi_{t,n}(e), \omega_{t,n}) \\ 0 & \text{other case} \end{cases}$$

- *Majority Vote Weighted by Leaf and by Tree to strategy 2* (MWLT2): Every reached leaf of the forest is considered to vote. In this case, given the example e, the forest votes for the class c if

  $$c = \max_{i=1}^{I} \sum_{t=1}^{T} p_t \sum_{n=1}^{N_t} K1(i, \chi_{t,n}(e), \omega_{t,n})$$

• **Weighted Implementations based in local fusion-** Within this group we define the following implementations:

- *Local fusion applied to strategy 1* (FUS1):

  To apply this combination method, first, during the learning of the forest, by each tree generated we obtain an additional tree, that we call error tree.

  The procedure to construct the error tree associated to t-th tree in the forest is the following:

  With the training data set of the t-th tree in the forest (tds tree$_t$ in figure 3) we make a test of the tree. So, we consider the training data set as the test data set. With the results of this test we build a new dataset (tds error_tree$_t$ in the figure 3) with the same data but replacing the attribute class for the binary attribute error. The attribute error indicates whether that example has been classified correctly or not by the t-th tree in the forest (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 is constructed a tree to learn the attribute error. This is the error tree.

  In the figure 3, tds tree$_t$ is the training dataset of the t-th tree in the forest and contains examples represented by vectors where:

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

  tds error_tree$_t$ is the training dataset of the error tree associated to the t-th tree in the forest. Contains vectors represented as:

  - $e_{j,t}$ is the j-th example in the training dataset of the t-th tree in the forest.
  - $error_{j,t}$ is the attribute that acts as class in this dataset. It's a binary attribute with value:
    - 1 if $e_{j,t}$ is incorrectly classified by the t-th tree in the forest.
    - 0 if $e_{j,t}$ is correctly classified by the t-th tree in the forest.

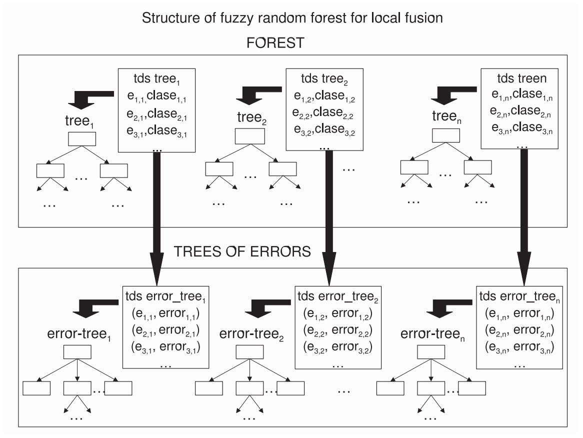

Figure 3: Inference with local fusion

Once the forest and the additional error trees are built, for each example e to classify and each tree of the forest we will obtain a vector $\overline{p}_e = (p_{e,1}, p_{e,2}, ..., p_{e,T})$ with the weights assigned to each tree to this example (local weights). Each $p_{e,t}$ is obtained as:

$$p_{e,t} = \sum_{n=1}^{N_{error_t}} \chi_{error_t,n}\omega_{error_t,n,0}, \quad \forall t = 1,..,T$$

where with the index $error_t$ we refer to the error tree of t-th tree in the forest, N_{error_t} is the number of leaves reached by the example e in the tree of errors $error_t$, $\chi_{error_t,n}$ is the weight of the leaf n of the tree of errors $error_t$ reached by the example and $\omega_{error_t,n,0}$ is the weight of the class 0 (value of binary attribute error=0) in the leaf n of the tree of errors.

The key idea that we want to capture with this implementation is that use of a local weight or a local fusion mechanism [3]. For a given probe (new example), first we evaluate the tree's performance with those examples (in the training set) which are similar to the probe we want to classify. Then, we associate a weight to the decision of that tree on the basis of its performance in the neighborhood of the probe.

Finally, the decision of the forest for the probe e is obtained by weighing the decisions of each tree with the computed weight. Therefore, the forest votes for the class c is:

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

where $A[t][i]$ is obtained using the majority vote 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:

$$A[t][i] = \begin{cases} 1 & \text{if } i = \max_{j=1}^{I} \sum_{n=1}^{N_t} K1(j, \chi_{t,n}(e), \omega_{t,n}) \\ 0 & \text{other case} \end{cases}$$

- *Local fusion applied to strategy 2* (FUS2): Every reached leaf of the forest is considered to vote. In this case, given the example e, the forest votes for the class c if

$$c = \max_{i=1}^{I} \sum_{t=1}^{T} p_{e,t} \sum_{n=1}^{N_t} K1(i, \chi_{t,n}(e), \omega_{t,n})$$

5 Experiments and results

In this section we will make the FRF inference with different databases of the UCI repository and with different percentages of missing values and linguistic labels (to introduce the missing values and linguistic labels we have used the NIP 1.5 tool [5]). The percentage is divided equally between missing and labels values. In table 1 show the results of four inference implementations that we are considering in this work for each database without imperfection, with 5%, 15% and 30% of missing and linguistic labels. For each of them show the average and the standard deviation of 5 x 10 cross validation. We also show the results of the combined outputs of the trees by simple majority vote applied to strategy 1 (SM1) where each tree votes for the majority class of the reached leaves and the forest votes for the majority class among the trees and simple majority vote applied to strategy 2 (SM2) where each reached leaf in the several trees of the forest votes for its majority class and the forest votes for the majority class among the reached leaves.

Analyzing data from Table 1 we can say that the four weighted inference implementations have have similar behavior but the best inference implementation is FUS2, followed by MWLT2, FUS1 and MLWT1. It seems that the weighted inference implementations behave better with the strategy 2 that the strategy 1 and within each one of them behave better those who make use of the weighted with local fusion. We may also note that although generally not weighted strategies (SM1 and SM2) have very good performance with databases without imperfection, weighted strategies have improved the not weighted when we add imperfection in the databases.

6 Summary

In this paper we have defined 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 but weighting the decision of each classifier by local fusion. The results of these combination implementation is compared with other weighted implementation based on the OOB dataset that obtain a good performance in FRF and implementations based on simple majority vote.

We have presented experimental results obtained by applying these implementations to various databases. The results show that all weighted implementations obtain a similar behavior but FUS2 is the implementation that gets better results. Also the results show that the general strategy 2 provide better results with these implementations than the strategy 1. Furthermore, the weighted strategies improves the non weighted strategies when we add imperfection in the databases.

Acknowledgements

The authors thank the “Ministerio de Educación y Ciencia” of Spain and the European Fund of Regional Development

Table 1: Testing accuracies of FRF for different levels of labels and missing

Dataset	Combination methods	original (0% missing and labels)	5% missing and labels	15% missing and labels	30% missing and labels
Ionosphere	FUS2	92.66 (2.15)	92.2 (2.16)	92.08 (1.96)	87.07 (3.29)
	FUS1	92.6 (2.11)	92.26 (2.17)	87.01 (3.36)	80.63 (2.96)
	MWLT2	92.66 (2.15)	92.2 (2.17)	91.51 (2.07)	85.82 (3.09)
	MWLT1	92.66 (2.15)	92.43 (2.17)	84.22 (3.01)	80.0 (2.77)
	SM2	93.0 (1.99)	92.14 (2.13)	85.93 (2.73)	79.83 (2.78)
	SM1	92.83 (2.05)	92.03 (2.15)	84.62 (2.82)	79.37 (2.81)
Wisconsin Breast Cancer	FUS2	96.37 (1.05)	95.94 (1.17)	96.02 (1.0)	94.79 (1.17)
	FUS1	96.34 (1.07)	95.62 (1.3)	95.54 (1.07)	94.22 (1.32)
	MWLT2	96.37 (1.05)	96.02 (1.15)	95.77 (1.04)	94.93 (1.11)
	MWLT1	96.34 (1.07)	95.65 (1.32)	95.34 (1.08)	94.05 (1.31)
	SM2	96.6 (1.03)	95.62 (1.32)	95.31 (1.07)	94.05 (1.31)
	SM1	96.6 (1.05)	95.62 (1.32)	95.31 (1.07)	94.05 (1.31)
Wine	FUS2	95.27 (2.35)	93.59 (2.62)	93.0 (2.75)	87.84 (3.85)
	FUS1	95.05 (2.49)	93.02 (2.81)	91.42 (3.14)	85.15 (4.56)
	MWLT2	95.27 (2.35)	93.47 (2.77)	92.78 (2.85)	87.61 (3.77)
	MWLT1	94.93 (2.57)	92.68 (2.91)	88.95 (3.27)	83.81 (4.43)
	SM2	95.27 (2.35)	92.01 (3.08)	88.39 (3.26)	83.58 (4.2)
	SM1	95.16 (2.45)	91.9 (3.12)	88.16 (3.36)	82.9 (4.5)
Iris Plants	FUS2	94.93 (3.13)	96.4 (2.5)	94.0 (3.2)	85.2 (5.13)
	FUS1	94.93 (3.13)	96.4 (2.5)	93.33 (3.2)	82.67 (5.67)
	MWLT2	94.93 (3.13)	96.4 (2.5)	94.0 (2.95)	85.33 (5.03)
	MWLT1	95.07 (3.16)	96.4 (2.5)	93.33 (3.2)	82.67 (5.71)
	SM2	94.93 (3.08)	96.4 (2.5)	93.33 (3.2)	82.4 (5.7)
	SM1	94.93 (3.08)	96.4 (2.5)	93.33 (3.2)	82.4 (5.7)
Pima Indian Diabetes	FUS2	76.59 (2.31)	73.36 (2.86)	72.11 (2.3)	72.42 (2.66)
	FUS1	76.51 (2.32)	73.02 (2.84)	71.72 (2.26)	72.61 (2.51)
	MWLT2	76.46 (2.34)	73.31 (2.9)	71.95 (2.31)	72.29 (2.83)
	MWLT1	76.66 (2.31)	72.89 (2.81)	71.8 (2.26)	72.16 (2.63)
	SM2	77.26 (2.23)	73.1 (2.58)	72.5 (2.38)	72.48 (2.76)
	SM1	77.0 (2.23)	73.54 (2.69)	72.4 (2.33)	72.5 (2.84)

for the support given to this work under the project TIN2008-06872-C04-03 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, M.J. Fazzari, N. Lim, J.J. Chen and R.L. Kodell. Classification by ensembles from random partitions of high dimensional data. *Computational Statistics and Data Analysis*, 51:6166–6179, 2007.

[2] P.P. Bonissone, J.M. Cadenas, M.C. Garrido and R.A. Díaz. Combination methods in a fuzzy random forest. *Proceedings IEEE International Conference on Systems, Man and Cybernetics (SMC2008)* 1794–1799, 2008.

[3] P.P. Bonissone, F. Xue, and R. Subbur. Fast meta-models for local fusion of multiple predictive models. *Applied Soft Computing Journal*, doi:10.1016/j.asoc.2008.03.06, 2008.

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

[5] J.M. Cadenas, J.V. Carrillo, M.C. Garrido, R. Martínez and E. Muñoz. A tool to handle imperfect information in data sets *URL: http://heurimind.inf.um.es*

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

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

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

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

[10] S. Segrera, M. Moreno. Multiclassifiers: applications, methods and architectures. *Proc. of International Workshop on Practical Applications of Agents and Multiagents Systems, IWPAAMS05*, 263–271, 2005.