A Fuzzy Random Forest: Fundamental for Design and Construction

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

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

M.C. Garrido
carmengarrido@um.es

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

Abstract

Following Breiman's methodology, we propose a multi-classifier based on a "forest" of randomly generated fuzzy decision trees, i.e., a *Fuzzy Random Forest*. This approach combines the robustness of multi-classifiers, the construction efficiency of decision trees, 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.

Keywords: Approximate Reasoning, Fuzzy Decision Trees, Random Forest, Combination Methods

1 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, [16].

In this paper, 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.

In section 2, we review the major elements that constitute the multi-classifier. In section 3, we explain the classic algorithm to create a random forest according to Breiman [3]. In the same section we also describe the adjustments, changes and considerations needed for the construction and inference of a fuzzy random forest. We present some initial results in section 4, followed by our conclusions in section 5.

2 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 [12].

Segrera [16] divides the methods for building multi-classifiers in two groups: ensemble and hybrid methods. The first type, such as Bagging [2] and Boosting [15], induces models that merge classifiers with the same learning algorithm, while introducing modifications in the training data set. The second, type such as Stacking [18], 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 [15], bagging [2] and random subspaces [7].

2.1 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 [14].

Breiman 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 [5], Forest-RI and Forest-RC [3], double-bagging [8].

Hamza [6] 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.

2.1.1 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 [9, 10, 11] 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.

Kuncheva [13] shows a comparative study of combination methods of fuzzy and non-fuzzy classifiers. Her work demonstrates that better results are obtained with fuzzy combination methods.

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 [9] 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. Fuzzy decision trees have two major components: a procedure for building fuzzy decision trees and an inference procedure for decision-making [11].

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 membership 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).

3 A Fuzzy Random Forest

The multi-classifiers proposed in this work are a forest of fuzzy decision trees generated randomly (Fuzzy Random Forest), following Breiman's methodology [3]. In this section we specify the adjustments, changes and considerations needed to construct this multi-classifiers .

3.1 Random Forest (following Breiman's Methodology)

The classic algorithm of learning and inference in a random forest according to Breiman, [3], is the following:

1. 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. Using the cases selected in the previous step, construct a tree (to the maximum size and without pruning). During this process, every time that it is needed to split a node, only consider a subset of the total set of predictor variables. Select the set of predictors as a random subset of the total set of available predictors. Perform a new random selection for each split. Some predictors (inclusive the best) cannot be considered for each split, but a predictor excluded in one split may be used by other splits in the same tree.
3. Repeat steps 1 and 2 to construct a forest, i.e. a collection of trees.
4. To score a case, run the example through each tree in the forest and record the predicted value. Use the predicted categories for each tree as "votes" for the best class, and use the class with the most votes as the predicted class.

Algorithm 1. Random Forest Algorithm

Random forests have two stochastic elements: 1) the selection of data set used as input for each tree; and 2) the set of predictor variables considered as candidates for each node split. These randomizations, along with combining the predictions from the trees, significantly improve the overall predictive accuracy.

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. To measure the generalization error of the forest, the OOB for each tree are run through the tree and the error rate of the prediction is computed. The error rates for the trees in the forest are then averaged to give the overall generalization error rate for the decision tree forest model.

There are several advantages to this method of computing generalization error: (1) all cases are used to construct the model, and none have to be held back as a separate test set; (2) the testing is fast because only one forest

has to be constructed (as compared to cross-validation where additional trees have to be constructed).

3.2 Approach and considerations to construct a Fuzzy Random Forest

In this work we propose to use Algorithm 1 to generate a random forest whose trees are fuzzy decision trees, proposing, therefore, a basic algorithm to generate a Fuzzy Random Forest (FRF).

Each tree in the forest will be a fuzzy tree generated following the guidelines of [10], 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

The fuzzy trees random generator follows Breiman's methodology:

1. Dividing the examples set of entry in subsets according to the step 1 of the algorithm 1.
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. This is done to be able to apply step 2 of algorithm 1.
3. Repeat steps 1 and 2 up to building all fuzzy trees. At the end, we will have constructed a fuzzy random forest.

Algorithm 3. FRF Learning

With this basic algorithm, 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.

3.3 Proposal and considerations for the inference in Fuzzy Random Forest

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 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 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.
- the weight $\omega_{t,n,i}$ if i is the majority class in this node and 0 for all other classes.

- the weight $\chi_{t,n}(e)$ if i is the majority class in this node and 0 for all other classes.
- 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.
- the weight $\omega_{t,n,i}$ for each class i.
- the weight $\chi_{t,n}(e) \cdot \omega_{t,n,i}$ for each class.

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

In Algorithm 4, we use the function $Faggre$, which is defined as a frequently used multi-classifiers combination method [12], 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, χ_{t,1}(e), ω_{t,1}), ...,
          K(1, χ_{t,N_t}(e), ω_{t,N_t}), ..., K(I, χ_{t,1}(e), ω_{t,1}),
          , ..., K(I, χ_{t,N_t}(e), ω_{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 4. 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, χ_{1,1}(e), ω_{1,1}), ...,
        K(i, χ_{1,N_1}(e), ω_{1,N_1}), ..., K(i, χ_{T,1}(e), ω_{T,1}),
        ..., K(i, χ_{T,N_T}(e), ω_{T,N_T}))
  End For each Class
End
```

Algorithm 5. FRF Inference (Strategy 2)

Now, we present some of the principal combination methods that we use to implement the $Faggre$ function.

• **Majority vote** assigns e to the class label most represented among the forest's trees outputs.

In strategy 1, the forest vote for the class k if:

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

where $A[t][i]$ is obtained using the same combination method but applied to the values of reached leaves in each tree:

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

In case of the second strategy every reached leaf of the forest is considered to vote. The forest will decide the class k if

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

□

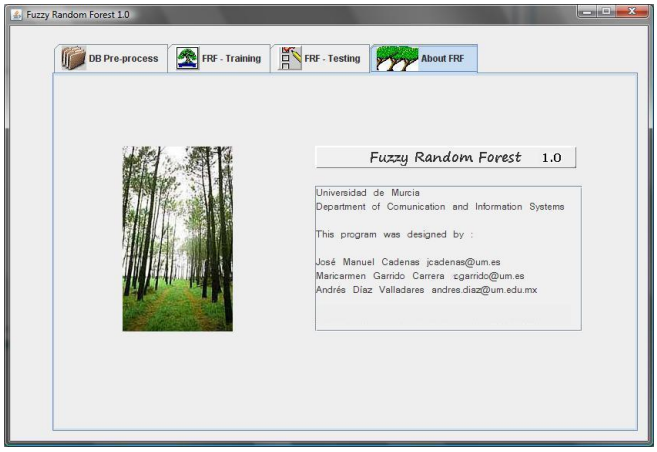

Figure 1: Screen of FRF 1.0

• **Minimum simple** combiner operates by taking the minimum in each class on leaves and trees.

In strategy 1, this combination method has 2 levels of application:

1) If we apply at decision level of every tree, in which case we have $A[t][i] = min(K(i, \chi_{t,1}(e), \omega_{t,1}), ..., K(i, \chi_{t,N_t}(e), \omega_{t,N_T}))$.

Then, take the most higher value on the vector as label suggested by classifier.

2) If the decision level is the forest, in which case we obtain the vector $(\mu_1, ...\mu_I)$, $\mu_i = min(A[1][i], ..., A[T][i])$.

Then, take the most high value on the vector as label suggested by classifier.

Strategy 2 is simpler to apply since it only has one decision level. In this case the vector $(\mu_1, ...\mu_I)$, is obtained as:

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

□

In a similar way, we can calculate the class support from the decision profile taking Maximum, Average and Product. Weighted versions of the simpler combination methods will be implemented taking into account aspects as the weight of reached leaves, standard error of each tree, the amount of imperfect information to construct each tree, etc.

4 Experiments and preliminary results

We are currently developing an application that is (1) capable of generating the random databases (Random Forest Generator), and (2) infer the classification of the forest (Fuzzy Classifiers). Figure 1 shows a screen of this application, which we have labeled FRF 1.0. We used FRF 1.0 to obtain our preliminary results. Among other databases, we used the Iris database from UCI and realized diverse tests. We constructed the fuzzy random forest and, considering the majority vote (strategy 1 and strategy 2), the behavior of the forest was similar or slightly better. In Table 1, we show a comparative data for classification of iris database from different techniques.

For this comparison we have used a set of techniques of platform Weka [17], FID3.4 [10] and the proposed multi-classifier FRF 1.0. We have used multiple versions of Iris database: the original database (without imperfection) and several versions the database with imperfection (linguistic labels). We have repeated these tests doing 10 fold cross-validation. Figures 2 and 3 show a fuzzy tree of the Fuzzy Random Forest FRF 1.0.

As it can be observed from Table 1, FRF 1.0 has a reasonably acceptable behavior, but with the advantage of the versatile management of information.

Table 1: Preliminary results from FRF 1.0

IRIS

Technique	without imperfection	10% linguistic labels	15% linguistic labels
Naives Bayes (NaiveBayes)	96.00 ± 4.42	—	—
C4.5 (J48)	94.00 ± 6.96	—	—
Neuronal Net (MultiplayerPerceptron)	97.33 ± 3.27	—	—
Ripple-Down-Rule (Ridor)	96.00 ± 4.42	—	—
Random Forest (RandomForest)	94.67 ± 5.37	—	—
Boosting (AdaBoostM1)	95.33 ± 5.85	—	—
Bagging	94.00 ± 5.70	—	—
FID 3.4	96.67 ± 3.33	96.67 ± 3.33	96.00 ± 4.42
FRF 1.0 (Strategy 1)	98.00 ± 4.27	96.67 ± 3.33	96.67 ± 3.33
FRF 1.0 (Strategy 2)	98.00 ± 4.27	97.33 ± 3.27	96.67 ± 4.47

5 Summary

This document presents the study of a multi-classifier system called *Fuzzy Random Forest* with a reasonably acceptable behavior. Further, the system has the advantages of uncertainty management and the comprehensibility of linguistic variables.

We have explained the underlying methodology and principal support techniques:

- We have presented a general description of a fuzzy random forest classifier.
- For *fuzzy trees random generator*, we realized a hybridization of the techniques of *random forest* and fuzzy trees for training
- For *fuzzy classifiers* we have presented the basic idea for the consideration of the individual classifications and how they are combined to obtain the joint classification. Also have distinguished some considerations that will contribute major accuracy to the classifiers.

We are currently building a fuzzy random forest prototype, with which we plan to validate our considerations to obtain efficient multi-classifiers with imperfect information.

Finally, we must clarify that throughout this presentation, we refer to the task of inference as classification, while the task of regression is implicit in this process. This is due to the fact that the numerical attributes are divided in linguistic labels and treated as nominal ones.

Acknowledgements

The work has been supported by the project TIN2005-08404-C04-02 ("Ministerio de Educación y Ciencia" of Spain and the European Fund of Regional Development).

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

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

[4] 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.

[5] 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.

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

[7] T.K. Ho (1998). The random subspace method for constructing decision forests.

| attributes=4 || candidates = 2 → $attrib_1$, $attrib_3$ || selection → $attrib_3$
| $attrib_3$ ∈Eti$_1$ with membership degree α_{31}: Set=38.00 Ver=0.00 Vir=0.00
| $attrib_3$ ∈Eti$_2$ with membership degree α_{32}: Set=0.00 Ver=32.49 Vir=0.00
| $attrib_3$ ∈Eti$_3$ with membership degree α_{33}: Set=0.00 Ver=14.51 Vir=6.00
| | attributes=3 || candidate attributes=2 → $attrib_1$, $attrib_4$ || selection → $attrib_4$
| | $attrib_4$ ∈Eti$_1$ with membership degree α_{41}: Set=0.00 Ver=14.51 Vir=0.00
| | $attrib_4$ ∈Eti$_2$ with membership degree α_{42}: Set=0.00 Ver=0.00 Vir=4.90
| | $attrib_4$ ∈Eti$_3$ with membership degree α_{43}: Set=0.00 Ver=0.00 Vir=1.10
| $attrib_3$ ∈Eti$_4$ with membership degree α_{34}: Set=0.00 Ver=0.00 Vir=4.00
| $attrib_3$ ∈Eti$_5$ with membership degree α_{35}: Set=0.00 Ver=1.00 Vir=39.00

Figure 2: A fuzzy tree of the Fuzzy Random Forest FRF 1.0 for IRIS

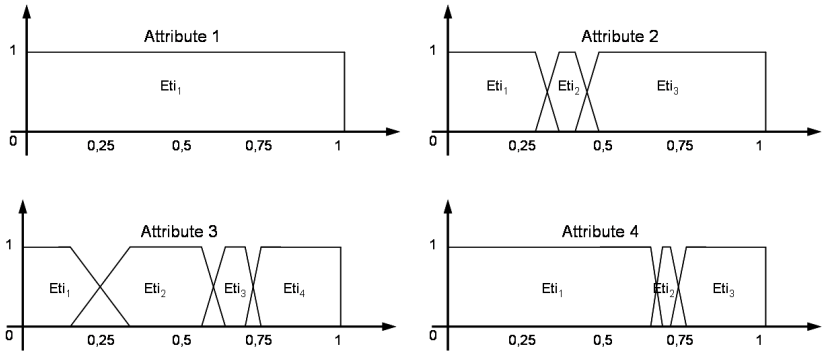

Figure 3: Attributes partition

Transactions on Pattern Analysis and Machine Intelligence 20(8), 832–844.

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

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

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

[11] 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.

[12] L.I. Kuncheva (2002). A theorical study on six classifier fusion strategies, *IEEE Transaction on PAMI* 24(2), 281–286.

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

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

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

[16] 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.

[17] University of Waikato Weka. Data Mining with Open Source Machine Learning Software in Java. URL: http://www.cs.waikato.ac.nz/ml/weka/

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