Expert Systems With Applications 154 (2020) 113436

Contents lists available at ScienceDirect

Expert Systems With Applications

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

An analysis of boosted ensembles of binary fuzzy decision trees

Marco Barsacchi, Alessio Bechini[*], Francesco Marcelloni

Dept. of Information Engineering, University of Pisa, Largo L. Lazzarino, Pisa 56122, Italy

ARTICLE INFO

Article history:
Received 18 June 2019
Revised 1 February 2020
Accepted 4 April 2020
Available online 10 April 2020

Keywords:
Ensemble classifiers
Boosting
Fuzzy characterization
Fuzzy decision trees
Rule-based classifiers

ABSTRACT

Classification is a functionality that plays a central role in the development of modern expert systems, across a wide variety of application fields: using accurate, efficient, and compact classification models is often a prime requirement. Boosting (and AdaBoost in particular) is a well-known technique to obtain robust classifiers from properly-learned weak classifiers, thus it is particularly attracting in many practical settings. Although the use of traditional classifiers as base learners in AdaBoost has already been widely studied, the adoption of *fuzzy* weak learners still requires further investigations. In this paper we describe FDT-Boost, a boosting approach shaped according to the SAMME-AdaBoost scheme, which leverages fuzzy binary decision trees as multi-class base classifiers. Such trees are kept compact by constraining their depth, without lowering the classification accuracy. The experimental evaluation of FDT-Boost has been carried out using a benchmark containing eighteen classification datasets. Comparing our approach with FURIA, one of the most popular fuzzy classifiers, with a fuzzy binary decision tree, and with a fuzzy multi-way decision tree, we show that FDT-Boost is accurate, getting to results that are statistically better than those achieved by the other approaches. Moreover, compared to a crisp SAMME-AdaBoost implementation, FDT-Boost shows similar performances, but the relative produced models are significantly less complex, thus opening up further exploitation chances also in memory-constrained systems.

1. Introduction

The automated support to decision-making provided by modern expert systems is often based on the ability to efficiently solve classification problems (Zelenkov, 2019). In many application fields, accuracy is not the only requirement for classification systems, but in resource-bounded platforms compactness is crucial as well: for example, it is the case of devices in IoT contexts (Dautov et al., 2018), and the wide range of embedded systems (Barry & Crowley, 2012), which for a long time are known to ask for computational power with constrained memory (Bechini, Conte, & Prete, 2004).

Decision tree can be regarded as one of the most popular classification techniques. Its widespread adoption is mainly due to its interpretability, and the simplicity of its supervised learning process has contributed to its success as well (Quinlan, 1986). Yet another positive aspect of decision trees is that the tuning of its learning process involves only a very limited number of parameters. The construction of a decision tree is driven by a recursive partitioning of the training dataset: A tree node corresponds to a subset of the whole dataset, and in turn the subsets of its children correspond to a partition of the father's data subset. Traversing the tree from the root down to the leaves, subsets become increasingly homogeneous with respect to the label associated with the contained examples.

Fuzzy representations have been developed to deal with inexact/uncertain information (Klir & Yuan, 1995). A prime motivation that drove the development of fuzzy methods was to construct *interpretable* systems; in fact, as fuzzy sets bridge the gap between the realm of numbers and the realm of concepts, they provide a way to represent knowledge in a linguistic and thus comprehensible way (Hüllermeier, 2011; Ricatto, Barsacchi, & Bechini, 2018). Other reasons often cited to support the usage of fuzzy set theory in machine learning are: i) the capability to incorporate background knowledge, ii) granularity (Pedrycz, Skowron, & Kreinovich, 2008), i.e. the capability of dealing with information granules at different levels of abstraction, and iii) graduality, the inherent capability of fuzzy systems to deal with gradual concepts. In the literature, in order to underline the difference between fuzzy and traditional approaches, models and algorithms that do not make use of any form of fuzziness are often indicated as "crispy," and in this work we shall adopt this terminology.

Fuzzy set theory found application in the field of decision trees through the proposal of *fuzzy decision trees* (FDTs) (Janikow, 1998). Differently from classical decision trees, a node in an FDT corresponds to a *fuzzy* set, and each instance can activate more than

[*] Corresponding author.
E-mail addresses: alessio.bechini@unipi.it (A. Bechini), francesco.marcelloni@unipi.it (F. Marcelloni).

https://doi.org/10.1016/j.eswa.2020.113436
0957-4174/© 2020 Elsevier Ltd. All rights reserved.

one branch, reaching multiple leaves. In general, fuzzy classification therefore leads to associate an instance with multiple classes, each with a different confidence value. As a consequence, FDTs have also been recognized as robust rankers (Hüllermeier & Vanderlooy, 2009).

Classifiers are said either *binary* or *multi-class*, depending on the number of categories considered in the target problem (two or more, respectively). In any case, the distinctive characteristic of a good classifier is its ability to provide accurate results: a proper learning process has a primary role in achieving high performances. Since Schapire found general results on the possibility to improve the performance of any mediocre classifier (Schapire, 1990), several approaches have been proposed to build one accurate predictive model by integrating multiple models (Rokach, 2010). In *ensemble methods*, a set of different *base learners* must be generated, and used for classification by collecting their predictions to determine a unique, agreed outcome; this approach has shown to deliver very satisfying results in many practical applications (see Ahmed, Rasool, Afzal, & Siddiqi, 2017; Hernández, Alonso, & Ocaña, 2017; Swiderski et al., 2017, just to mention some assorted recent ones).

Among ensemble methods, *boosting* plays a prominent role (Freund & Schapire, 1997; Schapire & Freund, 2012). Boosting is also known in the literature as "arcing", i.e. Adaptive Resampling and Combining (Breiman, 1998). In this approach, base classifiers are generated iteratively, and the base classifiers generated in previous iterations are used to properly manipulate the complete training set so as to derive the specific training set to be used in the next iteration (Schapire & Freund, 2012). Notably, boosting with decision trees as base learners has been soon considered as one of the most effective choices (Breiman, 1998). As decision trees are renown for their handiness, it is not surprising that they have often been chosen as base learners in boosting procedures (De'ath, 2007; Roe et al., 2005).

In the field of binary classification, AdaBoost is a popular boosting meta-algorithm known to be simple yet very effective (Freund & Schapire, 1997). It can operate with any kind of base learner (that is, obtained by any learning algorithm), and its tuning is based on one single parameter. Anyway, many real-world problems involve multiple classes, and the employment of AdaBoost in such contexts asks for specific algorithmic adaptations. Since the introduction of AdaBoost, the proposed multi-class boosting procedures have been shaped according to two main schemes (Saberian & Vasconcelos, 2011). In the first one, named *binary reduction*, a multi-class problem is recast to a combination of a number of binary sub-problems, each to be solved with the original AdaBoost. Unfortunately, binary reduction is subject to a number of problems (Zhai, Xia, & Wang, 2014). The second scheme plans to directly make use of native multi-class base classifiers, and use them in the context of a boosting procedure. Theoretical aspects of this scheme have been thoroughly investigated (Mukherjee & Schapire, 2013), and SAMME-AdaBoost (Hastie, Rosset, Zhu, & Zou, 2009) can be regarded as the most used method that follows this scheme. It is worth noticing that models generated by ensemble methods, as they are composed of several individual classifiers, may become really bulky unless each base component is kept compact. Anyway, empirical investigations have uncovered the interesting performances of boosted models with classification trees (Dietterich, 2000).

Recently, we proposed to use FDTs as base learners for a SAMME-AdaBoost procedure in Barsacchi, Bechini, and Marcelloni (2017). The very preliminary results obtained in this initial paper led us to the development of "FDT-Boost" classifiers, which are fully described in this paper for the first time, along with an experimental evaluation of several different aspects. In this context, the use of size-constrained binary FDTs bounds the complexity of the overall model. Moreover, an extensive assessment of the performance of the "FDT-Boost" classifiers allows us to understand their real potential. It is important to underline that a fair comparison with other models has to take into account not only the classification accuracy, but also the model complexity and the ability to deal with noisy data. To the best of our knowledge, no comprehensive analysis has been reported on this type of boosted classifier, and thus this work can fill the gaps in the current literature on the proper assessment of its potential in practical settings.

The prime contribution of this paper consists in the first detailed and comprehensive description of the FDT-Boost approach, along with an experimental investigation on the characteristics of its learning process and on its ability to deliver accurate and robust results, in a cross-comparison with other state-of-the-art fuzzy classifiers. Robustness in this context is intended as low sensitivity to label noise. In the evaluation of the learning process, a novel approach based on inspecting the trend of the weight entropy is proposed, along with more traditional ones. Another noteworthy result is presented: The complexity of models generated by FDT-Boost is lower than their non-fuzzy counterparts, created by using SAMME-AdaBoost with classical multi-class binary decision trees. This makes boosted models more suitable to be deployed in memory-constrained computing systems.

The paper is organized as follows. Section 2 recalls basic concepts on FDTs and fuzzy discretization. Section 3 is devoted to the description of the proposed FDT-Boost procedure. The results of the experimental assessment of FDT-Boost are reported in Section 4. Section 5 concludes the paper.

2. Preliminaries

For the sake of clarity, in the first place we briefly introduce basic notations. Each instance $\mathbf{x} = [x_1 \ldots, x_F]$ is characterized by a set $X = \{X_1, \ldots, X_F\}$ of F features (or "attributes"). The relative categorical label y takes values out of the set of M classes $\{C_1, \ldots, C_M\}$. The training set, containing N labelled instances, is indicated as $TR = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}$. It is worth underlining that in each iteration of the boosting procedure the actually used training set is obtained by sampling TR according to a given discrete distribution; the actual "sampled" training set used in the tth iteration is indicated in the following as $TR^{(t)}$. The notation $\hat{\mathbf{x}}$ refers to an unlabelled instance, to be possibly classified.

A decision tree implements a piece-wise continuous function in the feature space. Decision tree induction is obtained by deriving a proper hierarchical partition of the feature space, driven by the actual examples in the training set. Such a partition is obtained through a recursive approach: each set, corresponding to a node in the tree, is partitioned into k subsets, until a termination condition is met (Janikow, 1998). For $k = 2$, decision trees are said *two-way* (or *binary*), and for $k > 2$ they are said *multi-way*. Generalizations of this idea to the fuzzy context are used for FDT building as well (Altay & Cinar, 2016).

The overall learning procedure works on *categorical* attributes, and dealing with *continuous* attributes requires their discretization, to be performed either before or along the tree construction, as it happens for the well-known ID3 algorithm (Quinlan, 1986) and for CART (Breiman, 1993), respectively. Interestingly, Fayyad and Irani have shown that the application of entropy-based heuristics can lead to effective prior multi-interval discretization (Fayyad & Irani, 1993).

Along the decision tree construction, the splitting at a given node that generates the relative children is related to the local choice of a discriminating attribute. In the specific context of FDTs, the attribute selection rule may resort to different types of metrics; among them, the most frequently used are the fuzzy Kolmogorov–Smirnov discrimination quality measure (Boyen &

Wehenkel, 1999), the minimal ambiguity of a possibility distribution (Yuan & Shaw, 1995), the fuzzy Gini index (Chandra & Varghese, 2008), and the maximum classification importance of attribute contributing to its consequent (Wang, Yeung, & Tsang, 2001). Recently, the fuzzy information gain (Zeinalkhani & Eftekhari, 2014) has gained a widespread reputation.

The tree generation proceeds down a branch until a termination condition occurs. Usually, the termination condition is the logic disjunction of several *stopping predicates* that account for different kinds of imposed limitations, for example on the branch length, on the possible information gain, etc. Construction of binary and multi-way decision trees determines different properties for the final tree: in *multi-way* splitting of a node, one child is generated for any distinct linguistic value defined over the current splitting attribute. As a consequence, in multi-way DTs, no single attribute is tested twice on a path from the root to a leaf, while this is not guaranteed in binary DTs.

Because of the different nature of classical sets and fuzzy sets, the classification procedure is remarkably different in DTs and FDTs. In classical decision trees, each unlabelled instance $\hat{\mathbf{x}}$ corresponds to a unique leaf node (thus a unique class), reached traversing a unique path from the root downward. In FDTs, instead, $\hat{\mathbf{x}}$ can belong to different fuzzy sets with different membership degrees. In other words, from the root downwards, multiple paths are activated, getting to multiple leaf nodes. Specifically, the activation level of a leaf can be expressed by a *matching degree*. For a generic node $\mathcal{N}$ whose parent node is $\mathcal{PN}$, the matching degree $md^{\mathcal{N}}(\hat{\mathbf{x}})$ of instance $\hat{\mathbf{x}}$ with $\mathcal{N}$ is evaluated as

$$md^{\mathcal{N}}(\hat{\mathbf{x}}) = TN(\mu^{\mathcal{N}}(\hat{x}_f), md^{\mathcal{PN}}(\hat{\mathbf{x}})) \tag{1}$$

where TN is a T-norm, $\mu^{\mathcal{N}}(\hat{x}_f)$ is the membership degree of $\hat{x}_f$ to node $\mathcal{N}$ considering X_f as splitting attribute, and $md^{\mathcal{PN}}(\hat{\mathbf{x}})$ is the matching degree of $\hat{\mathbf{x}}$ to $\mathcal{PN}$. At each activated leaf node $\mathcal{LN}$, an instance $\hat{\mathbf{x}}$ can be associated up to a certain extent with a class C_m. To this aim, an *association degree* $AD_m^{\mathcal{LN}}(\hat{\mathbf{x}})$ is computed as

$$AD_m^{\mathcal{LN}}(\hat{\mathbf{x}}) = md^{\mathcal{LN}}(\hat{\mathbf{x}}) \cdot w_m^{\mathcal{LN}} \tag{2}$$

where $w_m^{\mathcal{LN}}$ is the *class weight* associated with class C_m at leaf node $\mathcal{LN}$. Class weights are used as they have been shown to improve performances in fuzzy classifiers (Ishibuchi & Yamamoto, 2005).

Indicating by G the set of training examples represented in the leaf node, and by G_{C_m} its subset of elements labelled with class C_m, the class weight $w_m^{\mathcal{LN}}$ is calculated as

$$w_m^{\mathcal{LN}} = \frac{|G_{C_m}|}{|G|}. \tag{3}$$

where $|\cdot|$ indicates the set cardinality. The actual output class label is finally obtained by the classifier by combining the *association degrees* for all the leaves in the FDT.

3. The FDT-Boost classifier

The learning of each base learner of the proposed FDT-Boost classifier starts with the fuzzy partitioning of continuous attributes, using a fuzzy entropy-based discretizer. The ensemble is fit together following the SAMME-AdaBoost algorithm. In the following subsections, all the basic procedures employed in FDT-Boost are described, namely the fuzzy discretizer, the construction of a weak learner, i.e. the fuzzy binary decision tree (FBDT), and SAMME-AdaBoost. The parameters whose values must be specified to drive the overall learning process (a.k.a. *hyperparameters*) are presented, and default values for them will be indicated later, as obtained also according to the outcome of specific empirical tests (see Section 4.1).

3.1. Fuzzy discretization

The discretizer is requested to directly determine fuzzy partitions of the domains of continuous attributes. Reworking a previously described solution (Segatori, Marcelloni, & Pedrycz, 2018), we have developed a fuzzy discretizer driven by evaluations of the fuzzy entropy associated with possible fuzzy partitions. The chosen fuzzy discretizer works on each continuous attribute by recursively defining strong triangular fuzzy partitions. This can be regarded as a fuzzy generalization of what has been proposed by Fayyad and Irani (1993). An important property of the chosen discretizer is its ability to perform *attribute selection*: in fact, considering the splitting of a target attribute domain, in case at the stopping point no partition has been generated yet, such an attribute can be removed.

In the explanation hereafter, we assume to work on a generic training set *TR*. It is intended that in the kth iteration of the boosting procedure, the sampled training set TR_k has to be used instead.

Let X_f be the *f*th attribute, and $x_{f,i}$ the value of X_f in the *i*-th sample in *TR*. We assume all the values $x_{f,i}, i = 1 \dots N,$ sorted in ascending order. Let I_f be a generic interval on the universe U_f of X_f, and let S_f be the relative set of examples in *TR* whose values for feature X_f fall in I_f. A fuzzy partition over I_f is denoted as $P_{I_f} = \{B_{f,1}, \dots B_{f,|P_{I_f}|}\}$, where $B_{f,p}$ is the *p*th fuzzy set defined over a subinterval of I_f, and $|P_{I_f}|$ is the number of fuzzy sets in the partition.

Each fuzzy set $B_{f,p}$ can be associated with the corresponding *support set* $S_{f,p}$: It is the subset of examples in *TR* that participate in $B_{f,p}$ or, formally, those whose values for the *f*th feature have a membership degree to $B_{f,p}$ strictly greater than zero:

$$S_{f,p} = \left\{ \mathbf{x} \mid \mathbf{x} \in TR \text{ AND } \mu_{B_{f,p}}(x_f) > 0 \right\}. \tag{4}$$

The number of elements of the support set, $|S_{f,p}|$, is known as *support value* or simply *support*, and the *fuzzy cardinality* $|B_{f,p}|$ of $B_{f,p}$ is defined as:

$$|B_{f,p}| = \sum_{j=1}^{|S_{f,p}|} \mu_{B_{f,p}}(x_{f,j}) \tag{5}$$

where $x_{f,j}$ is the value for the *f*th feature of the *j*th element in $S_{f,p}$.

The *fuzzy entropy* for a fuzzy set $B_{f,p}$ is defined as:

$$FEnt(B_{f,p}) = -\sum_{m=1}^{M} \frac{|B_{f,p,C_m}|}{|B_{f,p}|} \log_2 \left(\frac{|B_{f,p,C_m}|}{|B_{f,p}|} \right) \tag{6}$$

where $|B_{f,p,C_m}|$ is the fuzzy cardinality of $B_{f,p}$ restricted to just examples with class label C_m. The values of the fuzzy entropy for all the fuzzy sets that make up a fuzzy partition P_{I_f} over an interval I_f let us compute the relative *weighted fuzzy entropy* $WFEnt(P_{I_f}; I_f)$ as

$$WFEnt(P_{I_f}; I_f) = \sum_{i=1}^{|P_{I_f}|} \frac{|B_{f,i}|}{|S_f|} FEnt(B_{f,i}). \tag{7}$$

Two different partitions for the same interval I_f can be compared according to their weighted fuzzy entropy. In particular, we can quantify the effect of the substitution of a "base" partition P_{I_f} for another partition P'_{I_f} by means of the information gain defined as follows:

$$FGain(P'_{I_f}; I_f) = WFEnt(P_{I_f}; I_f) - WFEnt(P'_{I_f}; I_f). \tag{8}$$

A numeric feature is discretized by recursively operating on partitions of the relative complete range of values that is obtained considering all the examples in *TR*. The initial partition contains a unique "fuzzy" set with unitary constant membership. In a generic partitioning step of the discretization procedure, a

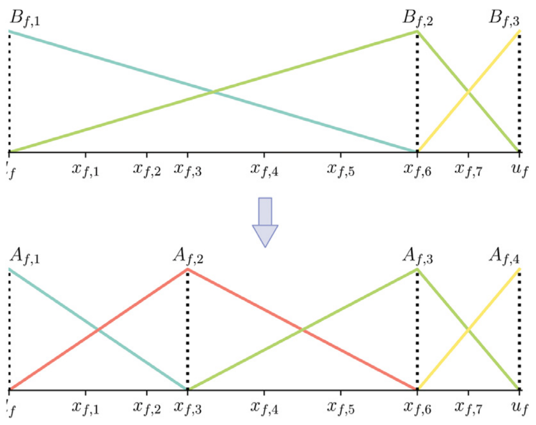


Fig. 1. An example of the application of the fuzzy discretization procedure for attribute X_f on $I_f = [l_f, u_f]$. First, the procedure selects $x_{f,6}$ as cut point, and generates the triangular fuzzy partition shown at the top. Then, it selects $x_{f,3}$ as cut point for the interval $[l_f, x_{f,6}]$ and generates the triangular fuzzy partition shown at the bottom.

sub-interval I_f (possibly) undergoes a partition made of three triangular fuzzy sets, which is fully described by an internal "cut-point" (see Fig. 1). The cut-point also separates two sub-intervals I_f^1 (left) and I_f^2 (right) of I_f, and the procedure is recursively applied to them. More specifically, the procedure operates on I_f along with its relative current partition P_{I_f}, by identifying possible cut-points and then choosing the cut-point that yields the further partition P'_{I_f} of I_f with the most favourable *FGain*. No further partition is applied in case the information gain falls below a given threshold (this is the stopping condition for the recursive procedure). Fig. 1 shows an example of two steps of the described procedure, on a given interval first, and then on its obtained left sub-interval.

The analytic form of the stopping condition is expressed as

$$FGain(P'_{I_f}; I_f) < \frac{\log_2(|S_f| - 1)}{|S_f|} + \frac{\Delta(P'_{I_f}; I_f)}{|S_f|} \tag{9}$$

where $\Delta(P'_{I_f}; I_f)$ is calculated as

$$\Delta(P'_{I_f}; I_f) = \log_2(3^{k_f} - 2) - \Big[k_{I_f} \cdot WFEnt(P_{I_f}; I_f) - k_{I_f^1} \cdot WFEnt(P'^{1}_{I_f}; I_f^1) - k_{I_f^2} \cdot WFEnt(P'^{2}_{I_f}; I_f^2)\Big] \tag{10}$$

where k_{I_f} is the number of class labels over the whole I_f, and $k_{I_f^1}$, $k_{I_f^2}$ are the numbers of class labels over I_f^1 and I_f^2, respectively.

Every time the stopping condition is immediately met at the inception of the discretization procedure for an attribute, we can deduce that such an attribute does not influence the output, and thus it will not be taken into account any more in the model construction.

In order to obtain a more effective discretization process by avoiding an unreasonable, excessive fragmentation of numerical attribute domains, the number of splits can be limited to a maximum s_{max}, i.e. constraining the number of generated fuzzy sets. Moreover, possible overfitting can be battled also by bounding the number of fuzzy sets. This can be obtained by tracing back the recursive splitting procedure, and preserving the top s_{max} splits in terms of fuzzy entropy gain, yet keeping the partitioning procedure consistent. This possibility is due to the additive formulation of entropy, and it needs only to keep track of the splitting tree; then the splitting tree can be traversed one step at a time, keeping at each iteration the node with the highest gain, among the available ones.

3.2. Construction of a base FDT

In the proposed ensemble boosted classifier, we decided to make use of a *binary* fuzzy decision tree (FDT) as weak base classifier. It is worth noticing that the multi-way FDT has not been deemed a good option mainly because it may exhibit problems in dealing with learning sets of limited size (Hastie, Friedman, & Tibshirani, 2013), whereas we intended to overcome this limitation. Moreover, a binary FDT has an additional advantage over a multi-way FDT: its complexity can be easily controlled by imposing the value of one single parameter (the maximum tree depth β).

In the proposed algorithm, an FDT is built according to a recursive node-splitting procedure that looks for the split that maximizes the relative fuzzy information gain, and is subject to a set of stopping conditions as well. The pseudocode for this learning algorithm is shown in Algorithm 1, and we shall refer to it in the following description.

ALGORITHM 1: Pseudocode of the tree construction procedure LearnFDT. As input, the algorithm requires the complete training set $\widehat{TR}$, the discretization $\mathbf{P}$, and the parameters β, n_{min}, γ, and η. The output is the FBDT.

```
Function LearnFDT($\widehat{TR}$, $\mathbf{P}$, $\beta$, $n_{min}$, $\gamma$, $\eta$):
    Create a root node with all the set of data $\widehat{TR}$, i.e. a fuzzy
    set of all the data, with all the membership values = 1.
    tree ← BuildTree(node, $\widehat{TR}$, $\mathbf{P}$, $\beta$, $n_{min}$, $\gamma$, $\eta$)
    return tree
Function BuildTree(node, $X$, $\mathbf{P}$, $\beta$, $n_{min}$, $\gamma$, $\eta$):
    if StoppingCondition(node, $\beta$, $n_{min}$, $\gamma$, $\eta$) then
        node ← mark node as leaf
    else
        splits ← SelectSplit($X$, $\mathbf{P}$)
        foreach $split_k \in splits$ do
            $child_k$, $X_k$ ← Membership($X$, $\mathbf{P}$, $split_k$)
            $child_k$ ← BuildTree($child_k$, $X_k$, $\mathbf{P}$, $\beta$, $n_{min}$, $\gamma$, $\eta$)
            connect $child_k$ with node
        end
    end
    return node
```

At the inception of the tree construction, a root node with all the data $\widehat{TR}$ is created. It corresponds to a fuzzy set containing all the data with a membership degree of 1. Then, the recursive BuildTree function is called on the root node. Once called, BuildTree verifies the stopping conditions over the current node. In our work, the chosen stopping predicates are:

- the node contains less than n_{min} fuzzy instances;
- the node contains examples of a class $\mathcal{C}_m$ with a proportion greater than or equal to a fixed threshold γ in the range $(\frac{1}{M}, 1)$;
- the fuzzy information gain for the best split is lower than a threshold η
- the maximum tree depth β has been reached.

If at least one of the above stopping conditions is satisfied, the node is marked as a leaf and the recursive branching is stopped. Otherwise, two child nodes are generated, choosing the split that maximizes the fuzzy information gain *FGain* which, in this case, has the same form indicated in Eq. (8), but referring to the partitioning of the node fuzzy set induced by the node split. The best split is found by means of the function SelectSplit, which iterates

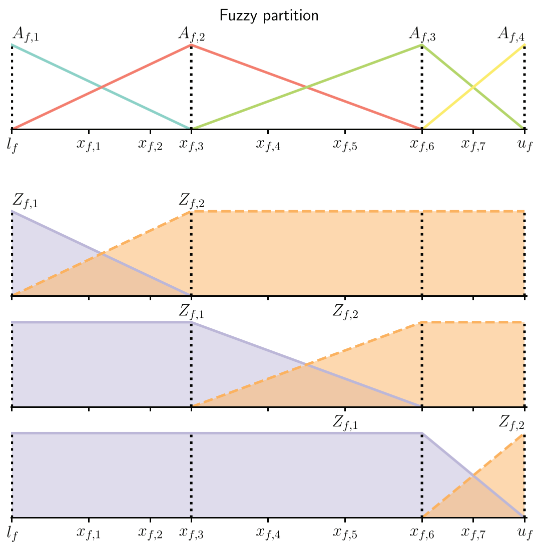


Fig. 2. An example of the application of the binary splitting procedure for attribute X_f on $I_f = [l_f, u_f]$. The procedure selects all possible candidates obtained by grouping together adjacent fuzzy sets into two disjoint groups and generates the trapezoidal fuzzy partitions shown at the bottom (Z_{f_1} and Z_{f_2}).

over all possible splits, evaluating *FGain* and returning the split with the highest gain.

To perform *binary* splitting with a partition **P** resulting from the discretization described in Section 3.1, it is necessary to evaluate $|\mathbf{P}| - 1$ candidates, with $|\mathbf{P}|$ as the number of fuzzy sets in **P**. Actually, each candidate is a binary fuzzy split made up by merging the first adjacent k fuzzy sets in **P** to form the first part, and all the remaining fuzzy sets to form the other. Both the parts are thus trapezoidal fuzzy sets. An example of all the candidate binary splits that correspond to a given partition is shown in Fig. 2, for the partition generated in the lower panel of Fig. 1.

Categorical attributes are instead dealt with according to a one-vs-all approach, which correspond to a binary split also in this case. This choice is motivated by the observation that, in general, no ordering is given on the possible category labels. Given a categorical feature $f_c \in C_f$, with $C_f = \{c_1, \ldots, c_{n_f}\}$, all the n_f different possible splits are evaluated; the ith candidate split separates c_i from all the other categories $C_f \backslash \{c_1\}$. It is worth noticing that, according to this choice, if only categorical features are present, the algorithm builds ups an ordinary binary decision tree. In the function SelectSplit, all the possible candidates are evaluated over all the attributes, and the best split made of the two fuzzy sets $Z_{f,1}$ and $Z_{f,2}$ is returned. Based on the outcome of SelectSplit, two child nodes, G_1 and G_2, are generated by calling the Membership function; it assigns to each node the examples belonging to the support of the respective fuzzy sets, with a membership value computed as the t-norm of the membership on the set and the membership on the father node. Following the indications in previous works with analogous membership calculation, in our implementation we used the product as t-norm.

3.3. Boosting via SAMME-AdaBoost

The boosting procedure used in the proposed classifier is aimed at generating the ensemble of base classifiers (namely binary FDTs) by following the SAMME-AdaBoost algorithm described in Hastie et al. (2009). SAMME-AdaBoost extends AdaBoost to the multi-class case, avoiding to reduce the classification problem to multiple two-class problems. It is worth recalling that SAMME just asks a base-learner to provide better predictions than an M-class random guess.

In the following we will refer to the overall procedure as *FDT-Boost*, and the relative pseudo-code is shown in Algorithm 2. We start with a base model, FBDT, a fixed number of iterations T, and constraints on the maximum number β of levels of the base model, the minimum number n_{min} of instances in a node, the minimum value γ of the proportion of examples of a class in a node, the minimum threshold value η for the fuzzy information gain, and the maximum number s_{max} (for the initial discretization). The base model, described in the previous sub-section, is grown up to its maximum depth β, with no post pruning. The optimal values for the parameter T and β depend on the characteristics of the specific problem; further details are given below (Section 4).

ALGORITHM 2: Pseudocode of the overall boosting procedure *FDT-Boost*. As input, the algorithm requires the complete training set *TR* and all the model parameters. The output is the ensemble of *FDTs*.

```
Function FDTBoost($TR$, $T$, $\beta$, $n_{min}$, $\gamma$, $\eta$, $s_{max}$):
    $\mathbf{P} \leftarrow$ FuzzyDiscretizer($\mathbf{X}$, $TR$, $s_{max}$)
    $BoostEns \leftarrow \varnothing$
    $\mathbf{w} = \{w_i \leftarrow 1/N, i = 1, \ldots N\}$
    /* Initially uniform sample weight vector */
    for $t \leftarrow 1$ to $T$ do
        $TR^{(t)} \leftarrow$ WeightedSampling($TR$, $\mathbf{w}$)
        /* $TR$ is sampled according to weight vector $\mathbf{w}$ */
        $FDT^{(t)} \leftarrow$ BuildTree($TR^{(t)}$, $\mathbf{P}$, $\beta$, $n_{min}$, $\gamma$, $\eta$)
        /* Construction of the $t$-th Fuzzy Decision Tree
            */
        $eer^{(t)} \leftarrow$ computation of Eq. 11
        $\alpha^{(t)} \leftarrow$ computation of Eq. 12
        /* Tree weight $\alpha^{(t)}$ depends on error rate $eer^{(t)}$ */
        $BoostEns \leftarrow BoostEns \cup (FDT^{(t)}, \alpha^{(t)})$
        $\mathbf{w} = \{w_i \leftarrow$ value as per Eq. 13, $i = 1, \ldots N\}$
        $\mathbf{w} \leftarrow \mathbf{w} / \sum_{i=1}^{N} w_i$
        /* Sample weight vector updated & normalized */
    end
    return $BoostEns$
```

Following the notation introduced before, the boosting procedure starts from considering the original training set with N samples, $TR = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}$, and one fuzzy partition (or "discretization") for each of the numerical attributes $\mathbf{X} = \{X_1, X_2, \ldots, X_F\}$. In fact, the fuzzy discretizer (Section 3.1) gives back **P**, a set of partitions for the attributes that have not been filtered out during the discretization process.

The requested set of T classifiers is built up by means of an iterative procedure. At each iteration t, the actual training set $TR^{(t)}$ to work on (still with N examples) is obtained by sampling TR according to the current sampling distribution, defined by its weight vector $\mathbf{w}$ with N components. At the first iteration, $TR^{(1)} = TR$. Out of a dataset $TR^{(t)}$ a base model $FDT^{(t)}$ is induced, and a weighted error rate $eer^{(t)}$ is computed as:

$$eer^{(t)} = \frac{\sum_{i=1}^{N} w_i \cdot \mathcal{I}(y_i \neq FDT^{(t)}(\mathbf{x}_i))}{\sum_{i=1}^{N} w_i} \tag{11}$$

being $\mathcal{I}(\texttt{true}) = 1$ and $\mathcal{I}(\texttt{false}) = 0$, and $FDT^{(t)}(\mathbf{x}_i)$ the classification output of the tth FDT for the ith sample in TR.

The contribution of the *t*th model $FDT^{(t)}$ to the boosted ensemble is accounted by the weight $\alpha^{(t)}$:

$$\alpha^{(t)} = \log \frac{1 - eer^{(t)}}{eer^{(t)}} + \log(M-1). \tag{12}$$

The rationale for the presence of the $\log(M-1)$ term (M indicates the number of classes) is the avoidance of negative values for α in the multiclass case for better-than-random guesses, as stated by the original SAMME algorithm (Hastie et al., 2009).

At each iteration all the sample weights are updated, increasing the values of samples that have been misclassified by the last built FDT:

$$w_i^{(t+1)} = w_i^{(t)} \cdot e^{\left[\alpha^{(t)} \cdot \mathcal{I}\left(y_i \neq FDT^{(t)}(\mathbf{x}_i)\right)\right]}, i = 1, \dots N. \tag{13}$$

A normalization across all the sample weights is then applied.

After the completion of the whole learning phase, the ensemble prediction for any unlabeled example $\hat{\mathbf{x}}$ is calculated taking into account all the votes from all the base learners in the ensemble. In the proposed fuzzy algorithm, each base $FDT^{(t)}$ produces as output, for each class C^m, a relative vote C_t^m. The ensemble-wide classification $Cl(\hat{\mathbf{x}})$ for $\hat{\mathbf{x}}$ consists in the single class that gets the highest cumulative vote, considering also the weights associated with the trees:

$$Cl(\hat{\mathbf{x}}) = \arg\max_m \sum_{t=1}^{T} \alpha^{(t)} \cdot C_t^m \tag{14}$$

At the end of the description of the overall approach, it is worth recalling some scalability issues, to be possibly addressed in future works. In the era of Big Data, designing scalable algorithms is a very important goal. Recently, several tree boosting approaches have been proposed to tackle the issues related to Big Data (Palit & Reddy, 2012). Anyway, because of its inherent sequential structure, the scalability of a distributed version of AdaBoost remains quite limited. Nevertheless, the implementation we have presented could benefit from employing a distributed FDT, with computation distribution at the node level (Segatori et al., 2018). In fact, with large datasets and allowing deeper depths for trees, the sequential nature of AdaBoost would hinder scalability only when the number of computing units becomes very large. The proposed system, mostly due to its modular nature, can be ported to a cluster computing framework. As the distributed version of both the discretizer and the fuzzy decision tree are available, and as AdaBoost encompasses the sequential section of the algorithm, we are confident that even more efficient distribution schemes could be designed and implemented, by exploiting possible synergy of different distributed versions of the composing parts.

4. Experimental investigations on FDT-Boost

The quantitative performance evaluation of classification algorithms is a very delicate task: for the sake of the reproducibility of results, we have only made use of datasets extracted from the Keel repository (http://sci2s.ugr.es/keel/datasets.php), because they are extensively used in the literature for the evaluation of learning methods (including the case of ensembles of decision trees Dietterich, 2000), and moreover they are publicly available also with pre-computed partitions according to 10/5-fold stratified cross validation (SCV) procedure (Alcalá-Fdez et al., 2011). In particular, for cross-validation purposes, we decided to make use of the pre-specified 5-fold splitting, as statistical evidence lets us have a proper size for each split over all the datasets (including those with a small number of examples), paired to a reasonable relative runtime. As FDT-Boost for categorical attributes practically behaves according to the ordinary AdaBoost scheme, datasets with categorical attributes only have been excluded from our benchmarks, deserving instead attention to those containing numerical attributes.

Table 1
Datasets in the “tuning benchmark” for FDT-Boost, and their characteristics.

Dataset	Cardinality	# Attr. (R,I,C)	# Classes
automobile	150	25 (15,0,10)	6
balance	625	4 (4,0,0)	3
bupa	345	6 (1,5,0)	2
ecoli	336	7 (7,0,0)	8
ionosphere	351	33 (32,1,0)	2
saheart	462	9 (5,3,1)	2
sonar	208	60 (60,0,0)	2

Table 2
Datasets in the “evaluation benchmark” for FDT-Boost, and their characteristics.

Dataset	Cardinality	# Attr. (R,I,C)	# Classes	*IR*
appendicitis (APP)	106	7 (7,0,0)	2	4.05
australian (AUS)	690	14 (3,5,6)	2	1.25
bands (BAN)	365	19 (13,6,0)	2	1.70
dermatology (DER)	358	34 (0,34,0)	6	5.55
glass (GLA)	214	9 (9,0,0)	7	7.78
hayes (HAY)	160	4 (0,4,0)	3	2.10
iris (IRI)	150	4 (4,0,0)	3	1.00
magic (MAG)	19,020	10 (10,0,0)	2	1.84
mammographic (MAM)	830	5 (0,5,0)	2	1.06
newthyroid (NEW)	215	5 (4,1,0)	3	5.00
ring (RIN)	7400	20 (20,0,0)	2	1.02
segment (SEG)	2310	19 (19,0,0)	7	1.00
tae (TAE)	151	5 (0,5,0)	3	1.06
vehicle (VEH)	846	18 (0,18,0)	4	1.07
vowel (VOW)	990	13 (10,3,0)	11	1.00
wdbc (WDC)	569	30 (30,0,0)	2	1.68
wine (WIN)	178	13 (13,0,0)	3	1.48
wisconsin (WIS)	683	9 (0,9,0)	2	1.86

For the performance evaluation of FDT-Boost we followed the widespread practice in the research literature on classification to refer to default parameter values for any algorithm involved in the experiments. For this reason, a first step has necessarily been the selection of reasonable default values for the hyperparameters of the proposed classifier. To this aim, we set up a first benchmark to be used in possible relative tests (we called it “tuning benchmark”). Moreover, an empirical cross-comparison with other algorithms requires the use of a dedicated benchmark as well (we denoted it as the “evaluation benchmark”). Thus, we assembled it with datasets not already used in the tuning benchmark. This last proviso guarantees that the default parameter values could not be considered as a benchmark-specific optimization in the cross-comparison tests. Actually, all the comparison algorithms are executed by using their suggested default values on the evaluation benchmark.

The datasets in both benchmarks have been chosen from the group “standard classification datasets” in the part of the Keel repository dedicated to supervised classification problems. The selection aimed to cover multiple aspects for general classification problems: different number of attributes, presence of real/integer features in different ratios, different sizes, difference balance levels. The final goal was to work with delimited yet representative subsets of a wide variety of classification problems possibly suitable for FDT-Boost, to be conveniently handled in computational tests. The chosen tuning benchmark is reported in Table 1, and the evaluation benchmark is reported in Table 2. Such tables report cardinality, number of attributes (real (R), integer (I), and categorical (C) ones), and the number of classes. Table 2 reports also the imbalance ratio *IR* (i.e. the number of instances of the most represented class over the number of instances of the less represented class - excluding the unrepresented ones).

In analyzing the performance of a classification system, it is important to refer to metrics that are able to properly characterize the target algorithms, and a wide assortment of metrics has been used in different settings described in the literature. In our experimentation, we aim to address classification problems that are reasonably balanced, with low-moderate imbalance ratios - a medium imbalance ratio *IR* is usually considered in the range 3–9 (Fernández, García, del Jesus, & Herrera, 2008). In this setting, accuracy is the most appropriate, synthetic, and intuitive index to quantify performance levels. As we can check in Table 2, all the selected datasets in the evaluation benchmark have low *IR* values, thus justifying the use of plain accuracy as reference metrics in our performance tests.

We present empirical results obtained in two distinct experiments. First, we focus on the comparison of FDT-Boost against other state-of-the-art classifiers that exploit the concept of *fuzziness*. Such first experiment is aimed at properly positioning our proposal in this set of models, which have been already compared with other popular ensemble classifiers (Hühn & Hüllermeier, 2009; Segatori et al., 2018) in the literature; the experiment is a classical benchmarking procedure. The comparison is carried out according to two techniques in subsequent tests. In the former, default parameter values are used for all the algorithms. In the latter, the obtained results are further substantiated by following a different methodological approach, performing a dataset-specific tuning of the models generated by each algorithm: a nested 5-fold cross-validation with validation and test set (Weiss & Kulikowski, 1991) is carried out, using inner 4-fold cross-validation for the selection of fold-specific hyperparameter values.

In the second experiment, we want to identify the actual benefits coming from the introduction of fuzziness in the corresponding "crisp" tree boosting scheme. Thus, we directly contrast FDT-Boost with SAMME-AdaBoost with regards to different aspects like accuracy, complexity, and robustness to label noise.

A crucial part of the proposed algorithm is the model learning phase, structured according to the AdaBoost approach. In order to better understand its behaviour, we analyze it with different conceptual tools, in the attempt to clarify the convergence properties on different datasets.

In the following subsection, we describe how default values can be chosen for the required hyperparameters; such parameterization is then used in the subsequent tests. In Section 4.2, the proposed fuzzy classifier is evaluated against other algorithms of the same category, i.e. fuzzy ones. Subsequently, the approach is compared also to the classical crisp SAMME-AdaBoost in Section 4.3, to possibly uncover the advantages derived from fuzzification. Section 4.4 is dedicated to the investigation of the convergence exhibited by the proposed model, studying also possible overtraining behaviours.

Our reference implementation of FDT-Boost has been developed in Python,[1] and it has been used in all the experiments described hereafter.

4.1. What standard values for hyperparameters?

The hyperparameters for the proposed model are summarized in Table 3, and they are relative to three different aspects: the discretization process, the induction of a fuzzy binary decision tree, and the boosting procedure. The default values presented in the table for these parameters are obtained as explained in the following.

We would like to point out that providing default values for these parameters allows employing the algorithm when a specific parameter optimization is not a reasonable option. In practical applications, of course, whenever we need to address one single specific problem, the algorithm can be tuned to work at its best by following a rigorous hyperparameter optimization procedure. Furthermore, optimization procedures might be hampered by computational issues, making the indication of general, default values very important. The selection of the required default values can be carried out by taking into account general results and experimental facts described in the literature. In cases where no indication is available, we resort to dedicated tests, obtained working on the "tuning benchmark" (see Table 1).

In the fuzzy discretization phase, the overall progression is regulated by Eq. (9), and only the maximum number of splits s_{max} has to be set. According to psychologists, to preserve interpretability, the number of linguistic terms per attribute should be 7 ± 2 due to a limit of human information processing capability (Miller, 1956). In our previous works, we experimentally verified that a value of 7 can be considered an upper bound, which guarantees a good trade-off between complexity and accuracy (Antonelli, Ducange, & Marcelloni, 2014). Although interpretability is not an objective when dealing with ensemble of classifiers, we adopted $s_{max} = 7$ since we observed in our experiments that higher values lead to excessive data fragmentation and model complexity, with no perceptible performance benefit (Antonelli et al., 2014).

More pre-training directions must be provided for the FDT induction. Fortunately, indications on default values n_{min}, γ and η can be found in the literature on fuzzy decision trees (Bechini, De Matteis, Marcelloni, & Segatori, 2016; Segatori et al., 2018). Conversely, the choice of the maximum tree depth β requires more investigation: β must be sufficiently high to guarantee a minimal error. Using the tuning benchmark, for each dataset, we decided to record the learning curves obtained varying β for iterations up to 1000, where the learning curve is expected to definitely show a plateau. Following established general results on tree boosting - see e.g. sect. 10.11 in Hastie et al. (2009) - we tested values for β in the range [1,6]. The results are shown in Fig. 3 as average curves for all the datasets. Clearly, the most convenient β must be selected along with a specific value for *T*.

Regarding the boosting procedure, the only parameter to study is the number of iterations *T*. It is worth recalling that the model complexity grows linearly with *T*, so it should be kept as low as possible. The value can be chosen by inspecting the learning curves for all the datasets in the tuning benchmark, spotting out the value where, on average, no relevant performance improvement is observed beyond. The choice of *T* depends also on β: observing the curves in Fig. 3, we can argue that a good trade-off between accuracy and model complexity is represented by $\beta = 4$ and $T = 500$. In fact, as in average no significant improvement in accuracy is obtained going beyond $\beta = 4$ (while the model complexity grows considerably), this value can be taken as a suitable one for general cases. On the other hand, for specific cases it might be recommendable using even lower values, which determine much simpler models.

Table 3
Values of default parameters for FDT-Boost.

	Discretization	Tree induction				Boosting
	$s_{\max}$	$n_{\min}$	γ	η	β	T
value	7	2	1.0	0.0001	4	500

4.2. FDT-Boost among fuzzy classifiers

In this section we compare FDT-Boost to some of the most significant state-of-the-art fuzzy classifiers. In the first place, we consider FURIA (Hühn & Hüllermeier, 2009). FURIA is an exten-

[1] Relative code made available by M. Barsacchi at https://bitbucket.org/mbarsacchi/fuzzyml/src/master/.

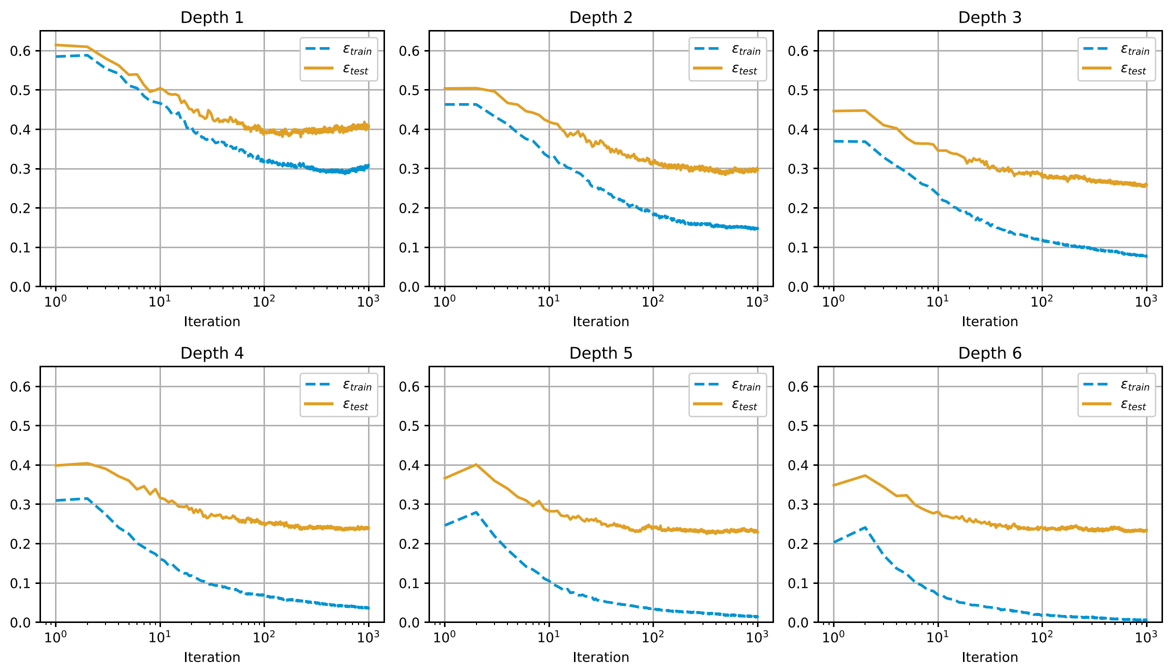


Fig. 3. Average classification error ratio vs. number of iterations over the tuning benchmark for different values of the maximum depth β for the base learner.

Table 4
Average accuracy and standard deviation achieved by FDT-Boost, FURIA, FBDT and FMDT over the test set (5-fold cross-validation) with standard values for hyperparameters.

Datasets	Algorithms			
	FDT-Boost	FURIA	FBDT	FMDT
APP	85.80 ± 7.43	**91.13** ± 0.75	83.07 ± 2.14	82.16 ± 8.43
AUS	**84.49** ± 0.35	83.30 ± 6.40	83.91 ± 1.34	84.20 ± 2.57
BAN	**74.51** ± 1.28	74.25 ± 1.97	64.89 ± 2.83	68.79 ± 2.63
DER	**97.78** ± 2.23	84.19 ± 4.87	94.69 ± 2.20	93.01 ± 1.88
GLA	74.31 ± 6.73	**84.95** ± 2.70	66.79 ± 5.22	71.95 ± 8.66
HAY	**85.00** ± 5.00	74.88 ± 2.54	73.75 ± 8.29	63.13 ± 6.67
IRI	**94.67** ± 2.49	93.33 ± 4.71	94.00 ± 3.89	94.00 ± 4.42
MAG	**85.56** ± 0.67	84.78 ± 0.52	79.59 ± 0.70	80.08 ± 0.72
MAM	**83.59** ± 1.63	79.01 ± 6.30	80.55 ± 2.00	80.44 ± 1.43
NEW	**94.88** ± 3.42	94.42 ± 2.37	93.02 ± 2.94	93.02 ± 1.47
RIN	**94.65** ± 0.78	83.69 ± 6.89	83.71 ± 6.81	87.04 ± 1.11
SEG	**96.80** ± 1.05	84.48 ± 4.67	96.79 ± 0.63	95.67 ± 0.51
TAE	**53.01** ± 7.48	45.63 ± 5.06	48.30 ± 7.71	51.61 ± 4.87
VEH	**72.21** ± 3.11	68.09 ± 1.78	70.92 ± 1.49	69.50 ± 1.71
VOW	86.46 ± 2.93	79.80 ± 2.09	79.60 ± 0.88	**94.44** ± 1.59
WDC	**97.18** ± 1.02	95.96 ± 2.12	94.38 ± 1.62	94.55 ± 2.03
WIN	**98.87** ± 1.38	93.78 ± 3.37	91.51 ± 5.73	94.38 ± 3.54
WIS	**97.01** ± 0.79	94.72 ± 7.18	95.32 ± 1.06	95.03 ± 1.23

Table 5
Results of the pairwise Holm post hoc test for significance level = 0.05 (approach with standard hyperparameter values).

Algorithm	Mean Rank	Iman-Davenport p-value
FDT-Boost	16.6667	0.000548992343
FURIA	41.5556	
FBDT	46.7778	
FMDT	41	

Table 6
Results of the Friedman Aligned statistical test on the accuracy obtained by FDT-Boost, FURIA, FBDT, and FMDT over the test set (approach with standard hyperparameter values).

Algorithm	z-value	*p*-value	Holm	Hypothesis
FBDT	4.316294	0.000016	0.016667	Rejected
FURIA	3.567711	0.00036	0.025	Rejected
FMDT	3.488075	0.000487	0.05	Rejected

sion of the RIPPER algorithm, which learns fuzzy rules instead of conventional rules and unordered rule sets rather than rule lists. Moreover, to deal with uncovered examples, FURIA makes use of an efficient rule stretching method. Then, we must observe that in empirical studies of boosting approaches it is reasonable, and also common practice, comparing the results of the boosted model with those of a full-featured version of the relative base classifier (Sun, Wang, & Wong, 2006). Thus, we take into account also FBDT, a fuzzy binary decision tree, and FMDT, a fuzzy multi-way decision tree (Segatori et al., 2018). Both FBDT and FMDT rely on a prior strong fuzzy partitioning of the domain of numeric attributes, and build up decision trees with no post-pruning. Information gain drives the splitting process: FBDT makes use of a binary splitting at each node, while FMDT takes into account all the composing partitions of an attribute at the same time. Even if FMDT models are more cumbersome than those of FBDT, on some datasets FMDT has been found to perform better (Segatori et al., 2018).

FURIA has been used as provided in WEKA (Eibe, Hall, & Witten, 2016) with its default parametrization (minimum number of covered instances $N = 2$; number of folds $F = 3$; number of optimizations $O = 2$), as suggested in Hühn and Hüllermeier (2009). Python implementations of FMDT and FBDT have been used, and their parameters have been set according to the indications reported in Segatori et al. (2018): the maximum depth has been fixed to 15 for FBDT, and 5 for FMDT.

A comparison can be initially carried out by referring to executions with standard parameters for all the involved algorithms, using 5-fold cross validation. The average accuracies (along with the relative standard deviations) obtained by the different algorithms on the test sets throughout the reference datasets are reported in Table 4. At first glance, the good performance of FDT-Boost is clear. Aiming to detect statistically significant differences between the approaches, we performed a set of tests. First, we defined the distribution with the average accuracies evaluated on the test set

Table 7
Accuracy results achieved by FDT-Boost, FURIA, FBDT and FMDT with dataset-specific model tuning.

Datasets	Algorithms							
	FDT-Boost		FURIA		FBDT		FMDT	
	Acc.	Δ_{Acc}	Acc.	Δ_{Acc}	Acc.	Δ_{Acc}	Acc.	Δ_{Acc}
APP	85.84 ± 7.38	+0.04	**90.38** ± 1.10	−0.75	87.75 ± 2.24	+4.68	84.07 ± 6.68	+1.91
AUS	**86.67** ± 1.87	+2.18	82.44 ± 6.99	−0.86	84.49 ± 1.98	+0.58	85.36 ± 3.41	+1.16
BAN	**75.31** ± 3.75	+0.80	73.26 ± 4.66	−0.99	65.17 ± 3.10	+0.28	68.55 ± 3.09	+1.70
DER	**98.06** ± 2.40	+0.28	81.70 ± 1.46	−2.49	81.70 ± 1.49	+0.55	93.30 ± 3.21	+0.29
GLA	75.70 ± 6.98	+1.39	**82.62** ± 3.94	−2.33	66.79 ± 5.22	0.00	72.39 ± 5.42	+0.44
HAY	**83.75** ± 4.59	−1.25	75.00 ± 2.53	+0.12	73.75 ± 4.24	0.00	63.75 ± 7.02	+0.62
IRI	94.67 ± 2.67	0.00	**98.00** ± 0.42	+4.67	96.00 ± 3.89	+2.00	94.00 ± 4.42	0.00
MAG	85.18 ± 0.65	−0.38	85.49 ± 0.12	+0.71	**85.63** ± 0.73	+6.04	80.03 ± 0.77	−0.05
MAM	**83.96** ± 1.58	+0.37	79.04 ± 6.32	+0.03	83.23 ± 2.39	+2.68	83.93 ± 2.46	+3.49
NEW	**96.74** ± 3.15	+1.86	97.67 ± 0.59	+3.25	93.02 ± 2.94	0.00	93.49 ± 1.70	+0.47
RIN	**95.54** ± 0.32	+0.89	83.70 ± 6.91	+0.01	92.55 ± 0.65	+2.18	91.28 ± 1.08	+4.24
SEG	**97.40** ± 0.64	+0,60	84.52 ± 4.80	+0.04	96.36 ± 0.57	-0.43	95.84 ± 0.86	+0.17
TAE	**52.32** ± 4.90	−0.69	48.74 ± 3.47	+3.11	49.01 ± 3.26	+0.71	51.61 ± 4.87	0.00
VEH	**74.11** ± 1.89	+1.90	71.09 ± 2.05	+3.00	71.40 ± 1.89	+0.48	70.33 ± 1.71	+0.83
VOW	**95.55** ± 1.25	+9.09	79.33 ± 1.86	−0.47	79.60 ± 1.94	0.00	94.44 ± 1.59	0.00
WDC	97.01 ± 1.43	−0.74	**98.42** ± 0.60	+2.46	94.38 ± 1.62	0.00	94.55 ± 1.18	0.00
WIN	**98.87** ± 1.38	0.00	98.43 ± 0.97	+4.75	91.51 ± 5.73	0.00	94.38 ± 2.52	0.00
WIS	**97.66** ± 0.84	+0.06	95.20 ± 7.34	+0.48	95.32 ± 1.06	0.00	95.03 ± 1.23	0.00

for all the datasets, and then we applied non parametric statistical tests.

Considering the specific setting and the best practices in performing non-parametric tests over data mining algorithms (García, Fernández, Luengo, & Herrera, 2010), we used the Friedman Aligned test for rankings the distributions: the ranks have been calculated referring to the averaged cross-validation accuracies, and are reported in Table 6. Moreover, we used the Iman and Davenport test (Iman & Davenport, 1980) to uncover statistical differences between the four distributions. In order to be marked as significant, the p-value must be lower than the significance level (for our experiments, we assumed the ordinary value 0.05); under this condition, the null hypothesis can be rejected and the existence of statistical differences among the distributions can be asserted.

According to the results shown in Table 6, FDT-Boost is associated with the lowest mean rank. Moreover, being $p\text{-value} = 0.010578 < 0.05$, the null hypothesis of statistical equivalence is rejected. Then, to uncover the pairs that show statistical differences, we used a Holm post hoc analysis: FDT-Boost is selected as control algorithm, because of its smaller rank. Post hoc results, shown in Table 5, clearly uncover how FDT-Boost outperforms the other approaches. Finally, it must be underlined that using the classical Friedman test (Friedman, 1937), the statistical conclusions are the same.

To further substantiate the results obtained so far, the comparison among classifiers has been carried out also according to a different methodology, aimed at tuning the hyperparameters of each algorithm on each specific dataset. Considering the small size of some datasets in our evaluation benchmark, the chosen approach has been a nested 5-fold cross-validation with validation and test set (Weiss & Kulikowski, 1991), with inner 4-fold cross-validation: different data are used to tune the model and to evaluate its performance, thus avoiding a bias to the dataset. In practice, to select hyperparameters in each outer cross validation iteration, the corresponding learning set is used for hyperparameter tuning purposes by means of a relative 4-fold inner cross-validation (i.e., keeping the same fold splitting). The best values in each inner iteration are obtained by a grid search. In each classification algorithm, the main hyperparameters that drive the model development have been taken into account: for FDT-Boost, the maximum depth β of the base binary tree and the number of iterations T, for FURIA the minimum number of covered instances N and the number of folds F, for FBDT the maximum depth β, and for FMDT the maximum depth β as well. Notably, the search space for FDT-Boost is quite wide. Regarding the grid search, the explored value ranges have been: for FDT-Boost,[2] $\beta \in [1, 7]$ with step 1 and $T \in [100, 1200]$ with step 100; for FURIA, $N \in [1, 4]$, $F \in [2, 4]$, and $0 \in [1, 3]$ with step 1, for FBDT $\beta \in [2, 18]$ with step 1, and for FMDT $\beta \in [2, 11]$ with step 1.

Table 8
Results of the Friedman Aligned statistical test on the accuracy obtained by FDT-Boost, FURIA, FBDT, and FMDT over the test set (approach with dataset-specific model tuning).

Algorithm	Mean Rank	Iman-Davenport p-value
FDT-Boost	19.6667	0.004902133188
FURIA	38.0833	
FBDT	45.4167	
FMDT	42.8333	

Table 9
Results of the pairwise Holm post hoc test for significance level = 0.05 (approach with dataset-specific model tuning.

Algorithm	z-value	p-value	Holm	Hypothesis
FBDT	3.691148	0.000223	0.016667	Rejected
FMDT	3.320838	0.000897	0.025	Rejected
FURIA	2.639947	0.008292	0.05	Rejected

The results of this additional test are reported in Table 7 and basically confirm the conclusions obtained using the standard hyperparameter values. In most of the cases, the different approach used in this comparison leads to moderately better accuracies than those obtained in the previous test, and such a difference for each value is indicated as Δ_{Acc} in Table 7. In FDT-Boost, for some datasets this approach is able to provide a significant improvement in performance with respect to the adoption of plain standard values (e.g. in the case of AUS, VEH, and VOW), and to build up less complex models. The statistical tests used previously have been applied also in this case, and their results (reported in Tables 8 and 9) still indicate the better behavior exhibited by FDT-Boost.

[2] Specific ranges have been used in FDT-Boost for datasets APP, AUS, HAY, and NEW ($\beta \in [2, 6]$ with step 1 and $T \in [25, 125]$ with step 25), and for MAG and RIN ($\beta \in [5, 9]$ with step 1).

Table 10
Comparison of FDT-Boost and crisp AdaBoost: average accuracy, and average number of nodes per base learner.

Datasets	Accuracy (%)		Number of nodes	
	FDT-Boost	AdaBoost	FDT-Boost	AdaBoost
APP	**85.80** ± 7.43	84.89 ± 2.08	**10.48**	20.56
AUS	84.49 ± 0.35	**86.66** ± 2.92	**7.93**	28.14
BAN	74.51 ± 1.28	**75.02** ± 3.53	**19.53**	26.54
DER	**97.78** ± 2.23	96.36 ± 1.04	**14.73**	17.23
GLA	74.31 ± 6.73	**77.10** ± 2.26	**25.66**	26.76
HAY	**85.00** ± 5.00	84.38 ± 5.08	**9.09**	20.08
IRI	**94.67** ± 2.49	91.19 ± 2.49	**7.85**	15.00
MAG	**85.56** ± 0.67	84.73 ± 0.63	3.44	19.38
MAM	**83.59** ± 1.63	78.63 ± 1.79	**1.88**	13.96
NEW	94.88 ± 3.42	**95.81** ± 2.27	19.78	**14.66**
RIN	94.65 ± 0.78	**94.66** ± 0.37	8.42	18.80
SEG	96.80 ± 1.05	**98.18** ± 0.34	27.82	**24.11**
TAE	53.01 ± 7.48	**55.03** ± 4.76	**15.15**	28.55
VEH	72.21 ± 3.11	**77.54** ± 2.42	29.02	**28.77**
VOW	86.46 ± 2.93	**91.12** ± 1.55	29.77	**28.24**
WDC	**97.18** ± 1.02	91.17 ± 1.53	**22.44**	24.56
WIN	**98.87** ± 1.38	95.46 ± 3.27	15.35	**15.00**
WIS	97.01 ± 0.79	**97.37** ± **0.94**	**16.47**	23.27

4.3. Advantages respect to crisp AdaBoost

One important aspect to investigate is how the usage of a *fuzzy* base learner, combined with a proper fuzzy discretizer, compares to a classical SAMME-AdaBoost approach, which makes use of a standard, “crisp” decision tree as weak learner (similarly, at the time of their proposal, fuzzy decision trees were compared to C4.5 (Chiang & Hsu, 2002)). To this aim, we ran both FDT-Boost and a SAMME-AdaBoost classifier on our evaluation benchmark. For both algorithms, the same maximum tree depth has been used ($\beta = 4$). The obtained results are shown in Table 10.

A quantitative comparison of the two different approaches reveals that, although both of them reach a very similar classification accuracy (see Table 12 for the statistical test), the FDT-Boost model is on average significantly less complex. Specifically, we evaluate the model complexity as the average number of non empty nodes across the whole ensemble (using the same maximum tree depth and number of iterations): This complexity index can be consistently used for both the systems. Table 10 indicates that FDT-Boost is, on average, ~ 30% less complex than SAMME-AdaBoost. We can conclude that the adoption of a fuzzy decision tree let us construct a more compact model, which may be more suitable for memory-constrained systems without compromising accuracy.

The ability to tolerate noisy data in the learning phase is recognized as a crucial characteristic, and specific not-boosted ensemble methods have been proposed to address it (Abellán, Mantas, Castellano, & Moral-García, 2018).

Fuzzy models are known to be able to deal with noisy data in a very satisfactory way (Chiang & Hsu, 2002; Klir & Yuan, 1995). On the contrary, the adaptation mechanism at the basis of AdaBoost is not expected to easily discriminate classification noise along the learning phase. Moreover, empirical results in the literature underline the good performances of boosting with ordinary decision trees, but not in noisy settings (Dietterich, 2000). For such reasons, we decided to compare also the ability to deal with noisy data of FDT-Boost and the classical SAMME-AdaBoost, in order to understand whether the employment of fuzzy base learners might mitigate the weakness of boosting in this aspect. In particular, we tested the robustness of the approaches at two different levels of label noise, corresponding to moderate and challenging noisy conditions. In all the experiments, we performed noise injection into the training set by randomly flipping the label values according to the tested noise level.

The outcomes of the overall comparison are reported in Table 11. We can see that, even if over several datasets FDT-Boost outperforms SAMME-AdaBoost, no hard evidence of its superiority can be found. Thus, the Wilcoxon test has been applied to uncover statistical differences among the distributions in Table 11, for both the settings. The test results, in terms of ranks and p-values, are reported in Table 12. In the moderate noise case, even if on most of the datasets FDTBoost performs slightly better, no statistical differences are found. Conversely, in the challenging noise case, FDT-Boost statistically outperforms its crispy counterpart, but only considering a 90% significance level. The trend of the ranking values indicates that the fuzzy approach becomes more suitable as more noise is injected.

In some particular cases we can note that SAMME-AdaBoost outperforms FDTBoost *in absence of noise*, but the situation is reversed at high noise levels. We decided to further investigate such cases, checking in detail what happens when the noise ratio is progressively increased, varying from 0% up to 35% with 5% increments. The detailed trend of error versus noise ratio for the Segment and the Wisconsin datasets is depicted in the charts in Fig. 4. On the Segment dataset - a quite large one - (see upper chart in Fig. 4) in the absence of noise the crisp model provides higher performance; anyway, as the noise ratio grows up the fuzzy model becomes increasingly more accurate and, with noise beyond 25%, it outperforms the other on the test set. On the Wisconsin dataset (lower chart in Fig. 4) the fuzzy approach turns to be very effective, outperforming the other as soon as a minimal amount of noise is present (less than 5%).

In summary, we can state in general that, as the noise level increases, the fuzzification inherent in an FDT base learner looks to slightly improve the boosting behaviour. Anyway, statistically relevant differences can be found only in challenging noisy settings.

4.4. Analysis of the model learning phase

Previous works in the literature have shown that boosting algorithms, although quite robust, may be prone to a growing generalization error in certain rare circumstances (Schapire & Freund, 2012). Indeed, it has also been pointed out that, in the case of AdaBoost, the ability to resist overfitting (Schapire, Freund, Bartlett, & Lee, 1998) wanes if the base weak classifier is too complex with respect to the dataset size, or if it is too weak (meaning that its margin against a random classifier is too small) (Schapire, 2013). As understanding possible generalization issues in our proposed boosting algorithm is of primary importance, we have carried out some experiments to investigate what happens when a fuzzy decision tree is chosen as base learner, considering that it is a more complex model than its crisp counterpart. In this section we carry out an analysis of the overall learning process, not necessarily limited to the assessment of possible overfitting. For the sake of clarity, our discussion explicitly refers only to two datasets that have been selected as representatives, but the charts that describe the analysis for all the reference datasets indicated in Table 2 are reported as supplementary material. “Vowel” has been chosen as a multi-class case where FDT-Boost is able to generate an effective model, experiencing no overtraining along its learning. Instead, “Mammographic” is one of the most complicated datasets considered here, where the proposed algorithm struggles even in learning examples in the training set. Despite of this, FDT-Boost outperforms the other algorithms on this dataset, arguably indicating the puzzling nature of the target problem.

Two different convergence patterns along the overall learning process are shown for the Vowel and Mammographic datasets in the upper rows of panels (a) and (b) in Fig. 5. A log scale is used for the x-axis. The charts in the upper row of panel (a) show the

Table 11
Average accuracy and standard deviation of FDT-Boost and SAMME-AdaBoost (on the test set) under different noise ratios.

Datasets	Moderate noise (10%)		Challenging noise (30%)	
	FDT-Boost	SAMME-AdaBoost	FDT-Boost	SAMME-AdaBoost
APP	**81.13** ± 6.99	80.22 ± 5.42	**76.36** ± 8.66	63.98 ± 13.22
AUS	**81.74** ± 4.01	81.01 ± 3.95	**74.64** ± 3.92	65.80 ± 4.52
BAN	**70.35** ± 3.39	67.99 ± 3.95	**60.62** ± 5.20	55.11 ± 4.94
DER	**93.89** ± 2.57	91.94 ± 3.98	**84.60** ± 4.23	83.00 ± 3.37
GLA	**71.96** ± 4.17	70.11 ± 3.88	**59.79** ± 6.29	57.97 ± 3.28
HAY	**76.25** ± 10.57	71.88 ± 5.59	58.75 ± 10.53	**60.62** ± 8.52
IRI	**86.00** ± 4.90	85.33 ± 7.48	**73.33** ± 12.14	71.33 ± 6.18
MAG	83.17 ± 0.61	**85.47** ± 0.50	78.49 ± 0.81	**82.75** ± 0.64
MAM	**84.06** ± 2.07	75.23 ± 2.02	**83.92** ± 2.16	64.94 ± 3.39
NEW	88.84 ± 7.97	**91.16** ± 4.74	**74.88** ± 8.70	74.42 ± 7.35
RIN	91.99 ± 0.58	**92.20** ± 0.49	**84.32** ± 0.95	78.42 ± 0.83
SEG	85.06 ± 2.05	**94.85** ± 0.81	**90.22** ± 2.79	86.62 ± 1.33
TAE	**51.68** ± 5.96	51.05 ± 6.56	**45.68** ± 7.41	43.83 ± 11.30
VEH	65.25 ± 2.71	**72.10** ± 1.72	62.06 ± 3.29	**64.30** ± 2.99
VOW	75.86 ± 2.38	**82.83** ± 2.91	60.80 ± 2.38	**66.87** ± 3.37
WDC	92.97 ± 2.71	**93.32** ± 2.21	**75.75** ± 3.08	74.52 ± 5.69
WIN	**94.35** ± 3.13	92.68 ± 3.39	75.79 ± 5.42	**75.87** ± 7.21
WIS	**95.76** ± 0.92	92.96 ± 1.55	**87.71** ± 2.60	80.67 ± 3.61

Table 12
Results of the Wilcoxon test for algorithm FDT-Boost, performed for the two different noisy settings.

vs. SAMME-AdaBoost	R^+	R^-	Exact P-value	Asymptotic P-value	Hyp. ($\alpha = 0.05$)	Hyp. ($\alpha = 0.1$)
Case: no noise	85.0	86.0	≥ 0.2	1	retained	retained
Case: moderate noise	98.0	73.0	≥ 0.2	0.571289	retained	retained
Case: challenging noise	129.0	42.0	0.05994	0.055338	retained	rejected

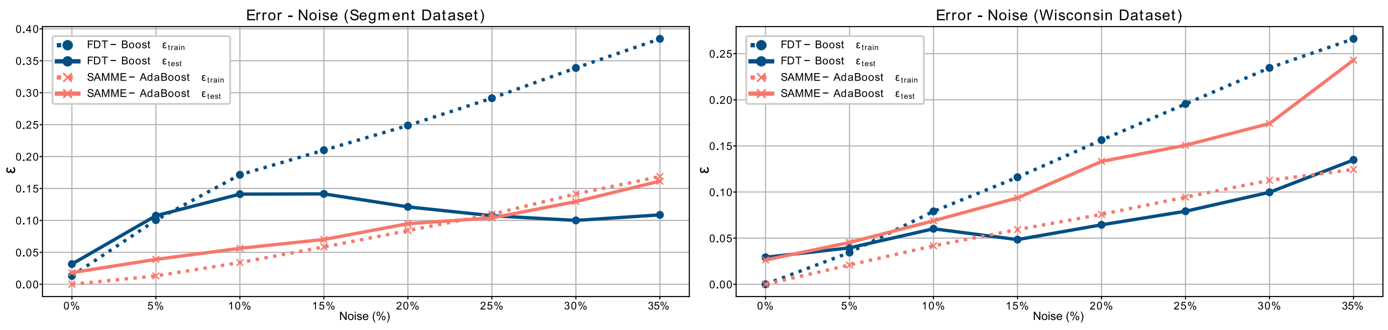
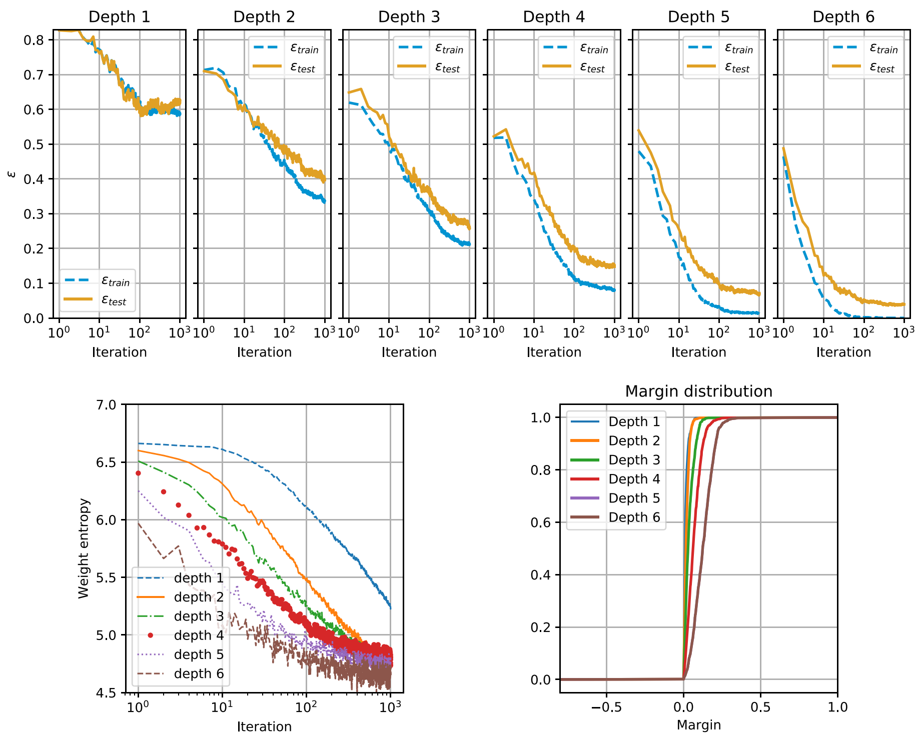


Fig. 4. The classification error on the Segment and the Wisconsin datasets, for both FDT-Boost and SAMME-AdaBoost, as a function of the percentage of noisy data. On these datasets, the noisier data are, the better the fuzzy model behaves with respect to its crispy counterpart.

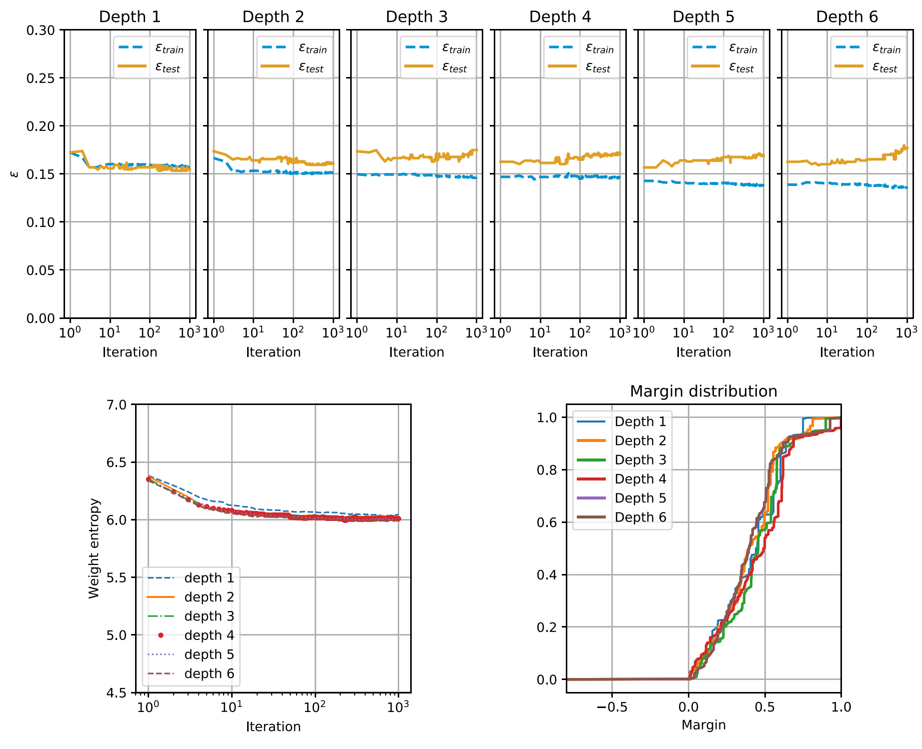
values for error ratio on both the training and test sets, as obtained in subsequent iterations on the Vowel dataset. In this specific case, as the number of iterations increases (in the chart, approaching 1000), the trend of the test error curve is monotonic descendent, just upon the training error curve, thus indicating that no overfitting occurs. It is evident that, by increasing the complexity of the base model, convergence is experienced with a lower number of iterations. On the contrary, an opposite behavior is witnessed in the case of the Mammographic dataset, shown in Fig. 5(b), with an increasing error ratio over the test set on the right part of the chart. Clearly, the fact that the training set can never be completely learned (ε_{train} never reaches the x-axis) impacts the actual classification performance of the learned model. Further, for this dataset we have found that a simpler base model is more robust with respect to overfitting. This result is in line with the general observation that the emergence of overfitting depends also on the complexity of the chosen base learner. These examples show how a dataset-dependent parametrization could definitively improve the accuracy. Even if a maximum depth of 4 provides a global tradeoff for all the datasets, it is a suboptimal choice in both the two cases shown here. Notably, for the Vowel dataset choosing $\beta = 6$ and $T = 600$ yields an average accuracy $ACC_{TS} = 96.26\%$, outperforming all the other algorithms (see Table 4). Likewise, with Mammographic, $\beta = 2$ and $T = 250$ let us slightly improve the achieved accuracy, getting up to $ACC_{TS} = 83.93\%$.

To better understand possible different behaviours with respect to convergence experienced in the learning procedure, we have performed two additional analyses: first, we have studied how the entropy of the weight distribution evolves along successive iterations; then, we have analyzed the margin distribution in the cases of trees with various maximal depths, according to the explanation of boosting proposed by Schapire et al. (1998). Considering our reference datasets, we have finally determined that most of them are not subject to overfitting under the proposed boosting approach. We verified that the problematic cases are only 3, namely Appendicitis, Australian, and Mammographic.

As shown in Fig. 5, the entropy of the distribution of sample weights is maximal at the beginning, where all samples are equally probable. Going on with the boosting iterations, the entropy decreases at a rate that depends on how fast the set of incorrectly

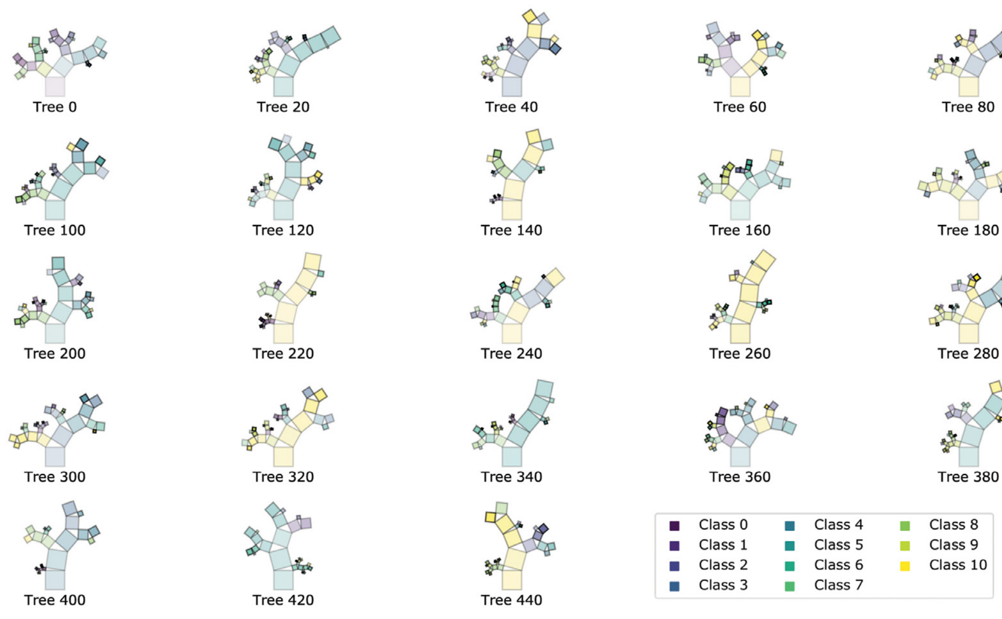


(a) Vowel

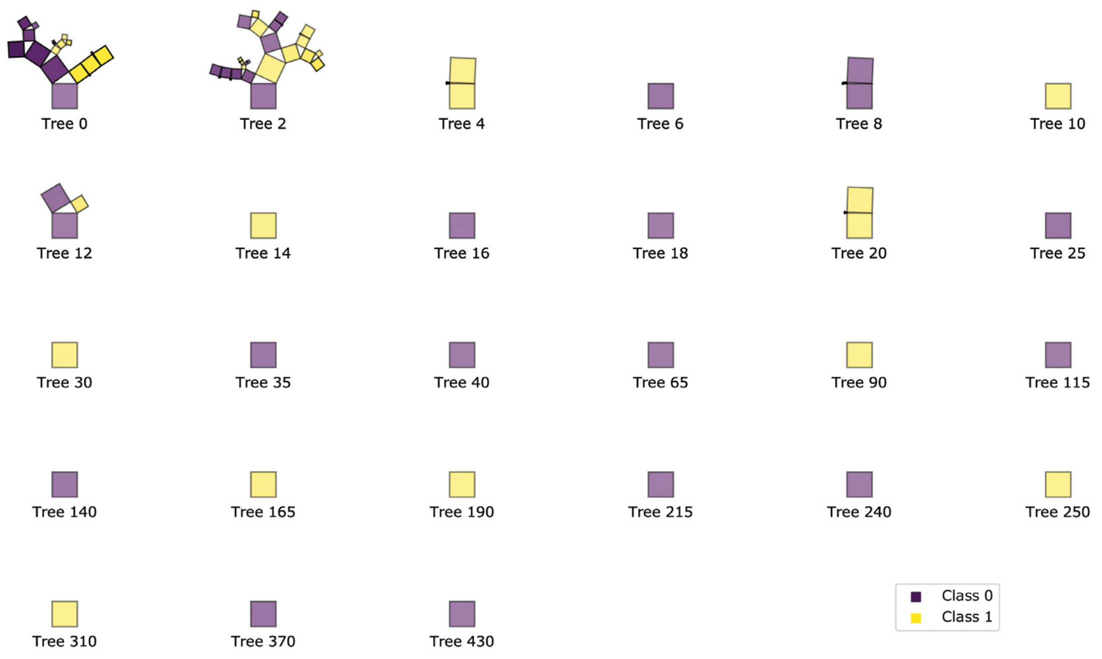


(b) Mammographic

Fig. 5. Results of the convergence analysis for the Vowel (a) and the Mammographic (b) datasets. The upper row reports the classification error ratio on the training and test sets (represented with dashed and continuous lines, respectively), as a function of the number of boosting rounds; the lower row shows the trend of the weight entropy, and the classification margin for different tree depths.



(a) Vowel



(b) Mammographic

Fig. 6. Some of the trees that make up the boosted ensemble trained on the Vowel (a) and Mammographic (b) datasets are shown by means of their Pythagoras tree representations. The two datasets lead to a different population of trees.

classified examples shrinks. This new perspective focuses on the evolution of the learning set $TR^{(t)}$, aiming to better understand how FDT-Boost works. In the lower left plot of panels (a) and (b) of Fig. 5 the entropy of the weight distribution is shown for various maximum tree depths as function of the iteration number. Again, for the Vowel dataset, a sharp decrease is shown for growing number of iterations, as well as for deeper base models. This is in stark contrast with the Mammographic case, which shows a milder decrease over iterations, even for deeper trees.

The classification margin for a sample is the difference between the sum of the weights assigned to the correct label and the maximal weights sum assigned to any single incorrect label. Here the margin assumes values in the range $[-1, 1]$ and a sample is classified correctly if and only if its margin is positive. Trivially, the larger the margin, the more confident the classification. Additionally, the margin value is influenced also by the number of classes, and this aspect must be taken into account in comparing the Vowel (11 classes) and Mammographic (2 classes) datasets. The *margin distribution graphs* in the lower left plots of panel (a) and (b) of Fig. 5 show the distribution of the margin over all the samples in the training set. For the Vowel dataset in panel (a), we can see that deeper base models help achieving better margin distributions, as well as better generalization error. On the contrary, for the Mammographic dataset, the margin distribution does not improve using more complex models, and thus the generalization error increases. It is worth underlining that however, despite the presence of the described overfitting, the best result across all the considered approaches is obtained by FDT-Boost.

Eventually, the convergence analysis for FDT-Boost turns out to be able to uncover the possibility to effectively carry out classification over the target dataset by means of sets of rules (decision trees, as long as FDTs, correspond to synthetic, structured representations of multiple rules). Thus, for example, rule-based classifiers are not particularly suited to deal with the Mammographic dataset. Anyway, also in this case the use of boosting helps in improving the classification accuracy (see the MAM row in Table 4).

Here, to provide a graphical effective viewpoint of the ensembles built up by the boosting procedure, in Fig. 6 we show a sample of the generated trees, making use of their Pythagorean representations (Beck, Burch, Munz, Di Silvestro, & Weiskopf, 2015). This portrayal draws each node as a square and each split as a right triangle, on the basis of the fact that the fuzzy cardinality of a sample in the parent node matches the sum of the fuzzy cardinalities in the child nodes. Furthermore, each node is colored according to the dominant class, and its transparency depends on the dominance margin.

According to Fig. 6(a), referring to the Vowel dataset, the different $TR^{(t)}$ sampled at each iteration show an oscillating dominance of examples from one of the leading classes. Thus, as the learning process goes on, the developed model becomes able to separate more difficult examples of the leading classes: trees become unbalanced and more specialized, being grown over more and more specialized training sets. The situation in Fig. 6(b), referring to the Mammographic dataset, is quite different: even if the different $TR^{(t)}$ becomes alternatively dominated by one of the two classes as for Vowel, trees fail to build up to the maximal depth, by mostly growing not beyond the root node after the very first iterations; in other words, they seem unable to separate the hardest examples. As shown in our experiments, Pythagoras trees have proven to be a valuable tool in understanding the evolution of the boosted ensemble. The evolution of the subsequently constructed base models can be effectively visualized, and countermeasures, such as earlier stopping or reduced tree depth β, can be put in place to get a better classifier.

5. Conclusion

In this work, we have presented a thorough description, as well as a comprehensive analysis of FDT-Boost, a novel ensemble-based fuzzy classifier. FDT-Boost is based upon a fuzzy discretizer for continuous attributes, and a depth-constrained binary fuzzy decision tree used as weak learner. The adopted boosting approach follows the SAMME-AdaBoost meta-algorithm. To the best of our knowledge, this is the most exhaustive investigation on the topic.

In our experiments we have found that SAMME-AdaBoost ensembles of binary fuzzy decision trees, limited to a maximum depth of 4, are typically able to provide high accuracy values. In fact, they produce results comparable to those obtained by state of the art fuzzy classifiers, and often even better ones. Indeed, on an evaluation benchmark with eighteen datasets, FDT-Boost outperformed FURIA, a fuzzy binary decision tree, and a fuzzy multi-way decision tree. Statistical analysis established those differences as significant. Such results have been confirmed also in a test with dataset-specific hyperparameter tuning in all the compared classifiers.

Furthermore, by contrasting FDT-Boost with its crisp counterpart, we have found that FDT-Boost produces much simpler, more compact models, with an average reduction of 30% in the number of nodes and leaves, reaching a comparable accuracy. This result is particularly important with regards to the possibility to exploit this type of classifiers on memory-constrained devices. In general, boosting solutions are not known for providing robustness against noise. With noise injection experiments we have seen that FDT-Boost behaves better than SAMME-AdaBoost only on specific datasets, and in particular in challenging noise settings; overall, modest statistically significant benefits due to the introduction of fuzziness can be found with 30% label noise.

Regarding the behavior of the learning phase of FDT-Boost, the convergence properties have been studied by means of different approaches, like margin analysis and entropy of weight distribution (a new tool proposed to understand the evolution of the boosting procedure along its iterations), showing that different datasets may lead to very different scenarios. Moreover, the model evolution has been also visualized using an effective graphical representation of the generated trees.

We observed that, as it often happens in similar cases, the final performances of the classifier substantially depend on the chosen parameter values. For this reason, future works will focus on methods to obtain a better automatic parametrization, taking into account the properties of the target dataset. Empirical investigations could be carried out to investigate the behaviour of the classifier over high-dimensional problems; moreover, possible ways to ascertain the interpretability of the learned model will deserve attention.

At the end of the description of the obtained results, it is worth recalling that the boosting scheme leverages the knowledge of previously built trees for the construction of the next tree: such a dependence determines scalability issues for any parallel implementation of the overall algorithm. Anyway, the algorithmic efficiency has to be pursued by studying any possible improvements to the scalability of the approach, to design a proper distributed version that could be able to take full advantage of the possibility to parallelize both the construction of the base classifiers and the progression of the boosting procedure. Solutions of this type will enable an easy application of the classifier to Big Data as well.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Credit authorship contribution statement

Marco Barsacchi: Conceptualization, Software, Investigation, Data curation, Writing - original draft. **Alessio Bechini:** Conceptualization, Investigation, Writing - original draft. **Francesco Marcelloni:** Conceptualization, Investigation, Writing - review & editing.

Acknowledgments

This work was been partially supported by the University of Pisa under grant PRA_2017 “IoT e Big Data: metodologie e tecnologie per la raccolta e l’elaborazione di grosse moli di dati” and by the Italian Ministry of Education and Research (MIUR) in the framework of the CrossLab project (Departments of Excellence). The implementation and evaluation of the described approach is part of the efforts for the projects “SIBILLA” and “TALENT,” co-financed by Regione Toscana under the framework POR-FESR 2014–2020 - Bando 2.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.eswa.2020.113436.

References

Abellán, J., Mantas, C. J., Castellano, J. G., & Moral-García, S. (2018). Increasing diversity in random forest learning algorithm via imprecise probabilities. *Expert Systems with Applications, 97*, 228–243. doi:10.1016/j.eswa.2017.12.029.

Ahmed, M., Rasool, A. G., Afzal, H., & Siddiqi, I. (2017). Improving handwriting based gender classification using ensemble classifiers. *Expert Systems with Applications, 85*, 158–168. doi:10.1016/j.eswa.2017.05.033.

Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., & Herrera, F. (2011). Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic & Soft Computing, 17*, 255?–287.

Altay, A., & Cinar, D. (2016). Fuzzy decision trees. In C. Kahraman, & O. Kabak (Eds.), *Fuzzy statistical decision-making: Theory and applications* (pp. 221–261). Cham: Springer International Publishing. doi:10.1007/978-3-319-39014-7_13.

Antonelli, M., Ducange, P., & Marcelloni, F. (2014). A fast and efficient multi-objective evolutionary learning scheme for fuzzy rule-based classifiers. *Information Sciences, 283*, 36–54. doi:10.1016/j.ins.2014.06.014.

Barry, P., & Crowley, P. (2012). *Modern embedded computing*. Boston: Morgan Kaufmann. doi:10.1016/C2011-0-05083-4.

Barsacchi, M., Bechini, A., & Marcelloni, F. (2017). Multi-class boosting with fuzzy decision trees. In *Proc. of 2017 IEEE int'l conf. on fuzzy systems (FUZZ-IEEE)*. IEEE. doi:10.1109/FUZZ-IEEE.2017.8015567.

Bechini, A., Conte, T. M., & Prete, C. A. (2004). Opportunities and challenges in embedded systems. *IEEE Micro, 24*, 8–9. doi:10.1109/MM.2004.30.

Bechini, A., De Matteis, A. D., Marcelloni, F., & Segatori, A. (2016). Spreading fuzzy random forests with MapReduce. In *Proc. of 2016 IEEE int'l conf. on systems, man, and cybernetics (SMC)* (pp. 2641–2646). IEEE Computer Society. doi:10.1109/SMC.2016.7844638.

Beck, F., Burch, M., Munz, T., Di Silvestro, L., & Weiskopf, D. (2015). Generalized pythagoras trees: A fractal approach to hierarchy visualization. In S. Battiato, S. Coquillart, J. Pettré, R. S. Laramee, A. Kerren, & J. Braz (Eds.), *Computer vision, imaging and computer graphics - theory and applications, chapter 8*. In *Comm. in Computer and Inf. Science: 550* (pp. 115–135). Springer. doi:10.1007/978-3-319-25117-2_8.

Boyen, X., & Wehenkel, L. (1999). Automatic induction of fuzzy decision trees and its application to power system security assessment. *Fuzzy Sets and Systems, 102*, 3–19. doi:10.1016/S0165-0114(98)00198-5.

Breiman, L. (1993). *Classification and regression trees*. Chapman & Hall.

Breiman, L. (1998). Arcing classifier (with discussion and a rejoinder by the author). *Ann. Statist., 26*, 801–849. doi:10.1214/aos/1024691079.

Chandra, B., & Varghese, P. (2008). Fuzzy SLIQ decision tree algorithm. *IEEE Transaction on Systems, Man, and Cybernetics, Part B: Cybernetics, 38*, 1294–1301. doi:10.1109/TSMCB.2008.923529.

Chiang, I.-J., & Hsu, J. Y. J. (2002). Fuzzy classification trees for data analysis. *Fuzzy Sets and Systems, 130*, 87–99. doi:10.1016/S0165-0114(01)00212-3.

Dautov, R., Distefano, S., Bruneo, D., Longo, F., Merlino, G., & Puliafito, A. (2018). Data processing in cyber-physical-social systems through edge computing. *IEEE Access, 6*, 29822–29835. doi:10.1109/ACCESS.2018.2839915.

De'ath, G. (2007). Boosted trees for ecological modeling and prediction. *Ecology, 88*, 243–251. doi:10.1890/0012-9658(2007)88[243:BTFEMA]2.0.CO;2.

Dietterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning, 40*, 139–157. doi:10.1023/A:1007607513941.

Eibe, F., Hall, M. A., & Witten, I. H. (2016). Appendix B: The WEKA workbench. In *Data mining: Practical machine learning tools and techniques* (p. 553?571). Morgan Kaufmann. doi:10.1016/B978-0-12-804291-5.00024-6.

Fayyad, U. M., & Irani, K. B. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. *Proc. of the 13th Int'l Joint Conference on Artificial Intelligence*, 1022–1027.

Fernández, A., García, S., del Jesus, M. J., & Herrera, F. (2008). A study of the behaviour of linguistic fuzzy rule based classification systems in the framework of imbalanced data-sets. *Fuzzy Sets and Systems, 159*, 2378–2398. doi:10.1016/j.fss.2007.12.023.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences, 55*, 119–139. doi:10.1006/jcss.1997.1504.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association, 32*, 675–701. doi:10.1080/01621459.1937.10503522.

García, S., Fernández, A., Luengo, J., & Herrera, F. (2010). Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences, 180*, 2044–2064. doi:10.1016/j.ins.2009.12.010. Special Issue on Intelligent Distributed Information Systems.

Hastie, T., Friedman, J., & Tibshirani, R. (2013). *The elements of statistical learning data mining, inference, and prediction* (2nd). Springer Verlag.

Hastie, T., Rosset, S., Zhu, J., & Zou, H. (2009). Multi-class AdaBoost. *Statistics and Its Interface, 2*, 349–360. doi:10.4310/SII.2009.v2.n3.a8.

Hernández, N., Alonso, J. M., & Ocaña, M. (2017). Fuzzy classifier ensembles for hierarchical wifi-based semantic indoor localization. *Expert Systems with Applications, 90*, 394–404. doi:10.1016/j.eswa.2017.08.007.

Hühn, J., & Hüllermeier, E. (2009). FURIA: An algorithm for unordered fuzzy rule induction. *Data Mining and Knowledge Discovery, 19*, 293–319. doi:10.1007/s10618-009-0131-8.

Hüllermeier, E. (2011). Fuzzy sets in machine learning and data mining. *Applied Soft Computing, 11*, 1493–1505. doi:10.1016/j.asoc.2008.01.004. The Impact of Soft Computing for the Progress of Artificial Intelligence.

Hüllermeier, E., & Vanderlooy, S. (2009). Why fuzzy decision trees are good rankers. *IEEE Transactions on Fuzzy Systems, 17*, 1233–1244. doi:10.1109/TFUZZ.2009.2026640.

Iman, R. L., & Davenport, J. M. (1980). Approximations of the critical region of the fbietkan statistic. *Communications in Statistics - Theory and Methods, 9*, 571–595. doi:10.1080/03610928008827904.

Ishibuchi, H., & Yamamoto, T. (2005). Rule weight specification in fuzzy rule-based classification systems. *IEEE Transaction Fuzzy Systems, 13*, 428–435. doi:10.1109/TFUZZ.2004.841738.

Janikow, C. (1998). Fuzzy decision trees: Issues and methods. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics), 28*, 1–14. doi:10.1109/3477.658573.

Klir, G. J., & Yuan, B. (1995). *Fuzzy sets and fuzzy logic: Theory and applications*. Prentice Hall.

Miller, G. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review, 63*, 81–97. doi:10.1037/h0043158.

Mukherjee, I., & Schapire, R. E. (2013). A theory of multiclass boosting. *Journal of Machine Learning Research, 14*, 437–497.

Palit, I., & Reddy, C. K. (2012). Scalable and parallel boosting with mapreduce. *IEEE Transactions on Knowledge and Data Engineering, 24*, 1904–1916. doi:10.1109/TKDE.2011.208.

Pedrycz, W., Skowron, A., & Kreinovich, V. (2008). *Handbook of granular computing*. New York, NY, USA: Wiley-Interscience.

Quinlan, J. (1986). Induction of decision trees. *Machine Learning, 1*, 81–106. doi:10.1023/A:1022643204877.

Ricatto, M., Barsacchi, M., & Bechini, A. (2018). Interpretable CNV-based tumour classification using fuzzy rule based classifiers. ACM. Proc. of the 33rd annual ACM symposium on applied computing, 54–59, 10.1145/3167132.3167135.

Roe, B. P., Yang, H.-J., Zhu, J., Liu, Y., Stancu, I., & McGregor, G. (2005). Boosted decision trees as an alternative to artificial neural networks for particle identification. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 543*, 577–584. doi:10.1016/j.nima.2004.12.018.

Rokach, L. (2010). Ensemble-based classifiers. *Artificial Intelligence Review, 33*, 1–39. doi:10.1007/s10462-009-9124-7.

Saberian, M. J., & Vasconcelos, N. (2011). Multiclass boosting: Theory and algorithms. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems 24* (pp. 2124–2132). Curran Associates, Inc..

Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning, 5*, 197–227. doi:10.1007/BF00116037.

Schapire, R. E. (2013). Explaining AdaBoost. In B. Schölkopf, Z. Luo, & V. Vovk (Eds.), *Empirical inference: Festschrift in honor of vladimir n. vapnik* (pp. 37–52). Springer Berlin Heidelberg. doi:10.1007/978-3-642-41136-6_5.

Schapire, R. E., & Freund, Y. (2012). *Boosting: Foundations and algorithms*. MIT press.

Schapire, R. E., Freund, Y., Bartlett, P., & Lee, W. S. (1998). Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics, 26*, 1651–1686. doi:10.1214/aos/1024691352.

Segatori, A., Marcelloni, F., & Pedrycz, W. (2018). On distributed fuzzy decision trees for big data. *IEEE Transaction on Fuzzy Systems, 26*. doi:10.1109/TFUZZ.2016.2646746.

Sun, Y., Wang, Y., & Wong, A. K. C. (2006). Boosting an associative classifier. *IEEE Transaction on Knowledge and Data Engineering, 18*, 988–992. doi:10.1109/TKDE.2006.105.

Swiderski, B., Osowski, S., Kurek, J., Kruk, M., Lugowska, I., Rutkowski, P., & Barhoumi, W. (2017). Novel methods of image description and ensemble of classifiers in application to mammogram analysis. *Expert Systems with Applications, 81*, 67–78. doi:10.1016/j.eswa.2017.03.031.

Wang, X.-Z., Yeung, D., & Tsang, E. (2001). A comparative study on heuristic algorithms for generating fuzzy decision trees. *IEEE Transaction on Systems, Man, and Cybernetics, Part B: Cybernetics, 31*, 215–226. doi:10.1109/3477.915344.

Weiss, S. M., & Kulikowski, C. A. (1991). *Computer systems that learn: Classification and prediction methods from statistics, neural nets, machine learning, and expert systems*. Morgan Kaufmann Publishers Inc..

Yuan, Y., & Shaw, M. J. (1995). Induction of fuzzy decision trees. *Fuzzy Sets and Systems, 69*, 125–139. doi:10.1016/0165-0114(94)00229-Z.

Zeinalkhani, M., & Eftekhari, M. (2014). Fuzzy partitioning of continuous attributes through discretization methods to construct fuzzy decision tree classifiers. *Information Sciences, 278*, 715–735. doi:10.1016/j.ins.2014.03.087.

Zelenkov, Y. (2019). Example-dependent cost-sensitive adaptive boosting. *Expert Systems with Applications*. doi:10.1016/j.eswa.2019.06.009.

Zhai, S., Xia, T., & Wang, S. (2014). A multi-class boosting method with direct optimization. In *Proc. of the 20th ACM SIGKDD int'l conf. on knowledge discovery and data mining, KDD '14* (pp. 273–282). New York, NY, USA: ACM. doi:10.1145/2623330.2623689.