Learning fuzzy decision trees using integer programming

Jason Rhuggenaath, Yingqian Zhang, Alp Akcay and Uzay Kaymak
School of Industrial Engineering
Eindhoven University of Technology
Eindhoven, The Netherlands
Email: {j.s.rhuggenaath, yqzhang, a.e.akcay, u.kaymak}@tue.nl

Sicco Verwer
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Delft, The Netherlands
Email: s.e.verwer@tudelft.nl

***Abstract*—A popular method in machine learning for supervised classification is a decision tree. In this work we propose a new framework to learn fuzzy decision trees using mathematical programming. More specifically, we encode the problem of constructing fuzzy decision trees using a Mixed Integer Linear Programming (MIP) model, which can be solved by any optimization solver. We compare the performance of our method with the performance of off-the-shelf decision tree algorithm CART and Fuzzy Inference Systems (FIS) using benchmark data-sets. Our initial results are promising and show the advantages of using non-crisp boundaries for improving classification accuracy on testing data.**

***Index Terms*—Fuzzy systems; Machine learning; Fuzzy logic; Optimization; Mathematical programming.**

I. Introduction

Decision trees or classification trees [1] are popular machine learning methods for supervised learning. Previous work by [2] has shown that the problem of learning optimal decision trees is an NP-complete problem. It is therefore not surprising that many popular decision tree algorithms (CART [1] and ID3 [3]) use greedy heuristics. There is also a stream of literature that uses concepts from fuzzy logic and fuzzy set theory in order to learn decision trees, see e.g, [4], [5] for a good overview. Popular greedy decision tree algorithms and the ideas behind them have also been adapted in order to incorporate these concepts, see for example [6] and [7].

In this paper we propose a new fuzzy decision tree method that learns decision trees of a given depth. We show that we can formulate the associated optimization problem as a Mixed Integer Linear Program (MIP). Previous work [8], [9], [10] and more recently [11], [12] has already considered such an approach for building crisp decision trees. The benefit of this approach is that one can take advantage of powerful MIP solvers and this is also reflected in the results in [11], [12]. To the best of our knowledge, we are the first to consider building fuzzy decision trees by formulating the learning problem as a MIP.

Existing fuzzy classifiers such as [7], [13], [14], [15] typically use heuristics to construct decision trees. One drawback of such an approach is that it is not clear whether there is substantial room for improvement on the returned solution. In our approach, we encode the decision tree learning problem into a MIP that can be passed to an off-the-shelf MIP solver such as CPLEX, which in principle can solve the problem to optimality. When the problem is too big to be solved to optimality in reasonable time, we can still gain information about the quality of the solution from the optimality gap that is returned from the MIP solver. Hence, our method is able to answer the question: how far is the learned decision tree from being optimal? Another advantage of our approach is that it provides extra flexibility. Similarly as in [12], constraints and objectives can be modified if one is interested in minimizing other objectives such as false positives, other than overall prediction errors.

Another popular approach for learning and predicting using fuzzy logic is to use a Fuzzy Inference System (FIS). Although there is software available that is easy to use (such as the fuzzy logic toolbox in MATLAB) and that can implement a FIS, they also have some limitations when applied to classification problems. Fuzzy Inference Systems tend to have good performance when the task is to predict a continuous outcome variable using continuous features, but typically, a FIS is not flexible enough to handle both categorical and continuous features as inputs at the same time. Furthermore, it is not clear how to adapt a FIS for multi-class classification problems. On the contrary, our method can handle both categorical and continuous features at the same time, and it can be used for multi-class classification problems.

We summarize the main contributions of this paper as follows:

- We propose the first encoding that formulates the problem of learning fuzzy decision trees as a mixed-integer linear program.
- The proposed encoding provides a general framework

that can handle both continuous and categorical data, and multi-class classification tasks. Therefore, our method is complementary to FIS and offers an alternative way for fuzzy modeling using decision trees that is suitable for situations in which a FIS cannot be applied.

- Experiments show that the performance of our method on training data is comparable to that of CART and FIS on most datasets. Interestingly, the learned model using our method seems to generalize better than CART an FIS, as shown by its performance on testing datasets.
- The proposed method is flexible since the objective function and constraints can be adjusted in a straightforward way depending on the specific prediction task (e.g. minimizing false positives) at hand. Moreover, different functional forms can be chosen for the membership functions.

The remainder of this paper is organized as follows. Section II discusses our proposed methodology for learning optimal fuzzy decision trees using integer programming (FDTIP). The formulated problem can be directly solved by any MIP solver such as CPLEX. In Section III, we compare the performance of our method with the performance of off-the-shelf decision tree algorithm CART and Fuzzy Inference Systems (FIS) using several real datasets. Section IV concludes our work and provides some interesting directions for further research.

II. Fuzzy Decision trees using integer programming

We assume that the reader has a basic understanding of decision trees. More information can be found in [16]. Our approach for learning fuzzy decision trees consists of three methodological steps. In the first step the raw features from the dataset are transformed using membership functions to membership values in the interval $[0, 1]$. In the second step we formulate the optimization problem associated with the learning of the fuzzy decision tree as a MIP. In the third step we evaluate the learned tree using several defuzzification methods. Fig. 1 gives an overview of our proposed approach.

A. Feature transformation using membership functions

We assume that there are n rows in the dataset and the dataset contains m (raw) features. Denote the class or label of data row r by $t(r)$. We assume that the initial dataset contains three types of features: continuous features, categorical features and binary features. We transform the categorical features into binary features by using one-hot encoding. Each raw feature i is associated with k_i fuzzy sets via its corresponding membership function. Our fuzzy decision tree operates on the membership values of the $\sum_{i=1}^{m} k_i$ fuzzy sets. That is, the actual features in our model are the membership values of the fuzzy sets and not the raw features themselves.

For each raw feature i, let $\mu_{i,j}(x),\ j = 1, \ldots, k_i$ denote the membership function of fuzzy set j. The interpretation of fuzzy set j depends on whether the feature i is continuous or not. If feature i is categorical, then k_i equals the number of categories and $\mu_{i,j}(x)$ is equal to the binary feature obtained from the one-hot encoding. If feature i is binary, then $k_i = 1$ and $\mu_{i,j}(x)$ is equal to the original binary feature.

In our decision tree, the fuzzy sets associated with each continuous feature represent intervals. More precisely, for each continuous feature i, the membership function $\mu_{i,j}(x)$ indicates the degree to which the value of feature i is as least as large as a threshold value $c_{i,j}$. That is, $\mu_{i,j}(x)$ gives the membership value of feature i to the set $F_{i,j} = \{x|\ x \geq c_{i,j}\}$ at the point x. For example, $\mu_{i,j}(v_i) = 0.7$ indicates that the value of feature i given by v_i satisfies $v_i \geq c_{i,j}$ with degree 0.7. Similarly, $1 - \mu_{i,j}(x)$ gives the membership value of feature i to the set $F'_{i,j} = \{x|\ x < c_{i,j}\}$ at the point x.

In our method, we used Random Forests [17] in order to generate multiple candidate values of $c_{i,j}$ for each continuous feature. The main idea behind this approach, is that the Random Forests (RF) will detect the boundary of critical regions in the feature space that are important for predicting the outcome variable in an efficient way. This similar to the approach used in [18], [19], [20] where boundaries obtained from a crisp decision tree learned by CART or ID3 are fuzzified in order to derive fuzzy rules. In [18] a crisp decision tree learned by CART is used to set up an initial structure of a Fuzzy Inference System (FIS). However, instead of using only one tree as in [18], we use a forest consisting of N_T trees in order to generate candidate values of $c_{i,j}$.

It could happen that some continuous features are not selected by any of the trees in the RF. For those features, we set the values of $c_{i,j}$ equal to the values of N_G equally spaced points in the range of the feature space. For example, if feature i has range $[0, 10]$, then setting $N_G = 5$ would give the following thresholds $c_{i,1} = 0, c_{i,2} = 2.5, c_{i,3} = 5, c_{i,4} = 7.5, c_{i,5} = 10$.

Not all of the thresholds generated by the N_T trees are informative. For a continuous feature i it may happen that multiple thresholds separate the same data rows in the dataset. More specifically, let $v_{i,r}$ denote the value of feature i in row r of the dataset and let $\mathcal{C}_i$ denote a set of thresholds for feature i generated by the RF. Then it might be that $M_t = |\{x|x \in \{1, \ldots, n\}, v_{i,x} \geq t\}|$ is the same for all $t \in \mathcal{C}_i$. In that case, any convex combination of the thresholds in $\mathcal{C}_i$ would also separate the same rows in the dataset. In our experiments, we only use the median of the thresholds in $\mathcal{C}_i$.

In our experiments we used S-shaped and Z-shaped membership functions for $\mu_{i,j}(x)$. The S-shaped membership function is given by:

$$\mu^S_{i,j}(x) = \begin{cases} 0 & \text{if } x \leq a \\ 2\left(\frac{x-a}{b-a}\right)^2 & \text{if } a \leq x \leq \frac{b+a}{2} \\ 1 - 2\left(\frac{x-b}{b-a}\right)^2 & \text{if } \frac{b+a}{2} \leq x \leq b \\ 1 & \text{if } x \geq b \end{cases}$$

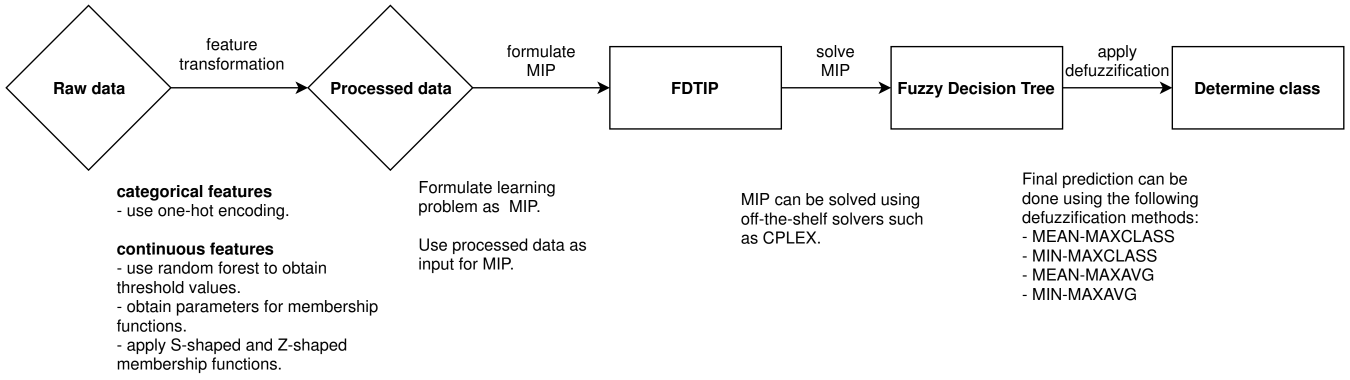
Raw data
feature transformation
Processed data
formulate MIP
FDTIP
solve MIP
Fuzzy Decision Tree
apply defuzzification
Determine class
categorical features
- use one-hot encoding.
continuous features
- use random forest to obtain threshold values.
- obtain parameters for membership functions.
- apply S-shaped and Z-shaped membership functions.
Formulate learning problem as MIP.
Use processed data as input for MIP.
MIP can be solved using off-the-shelf solvers such as CPLEX.
Final prediction can be done using the following defuzzification methods:
- MEAN-MAXCLASS
- MIN-MAXCLASS
- MEAN-MAXAVG
- MIN-MAXAVG

Fig. 1. Overview of proposed approach.

The Z-shaped membership function is given by:

$$\mu_{i,j}^{Z}(x) = \begin{cases} 0 & \text{if } x \leq a \\ \frac{x-a}{b-a} & \text{if } a \leq x \leq b \\ 1 & \text{if } x \geq b \end{cases}$$

The S-shaped and Z-shaped membership functions take two parameters a and b that locate the extremes of the function. If feature i was selected by RF with associated threshold value $c_{i,j}$, then we used $a = \text{median}\{x|x \leq c_{i,j}\}$ and $b = \text{median}\{x|x \geq c_{i,j}\}$. By choosing a and b in this manner, the extremes are determined in a data-driven way depending on the distribution of feature i around the threshold value $c_{i,j}$. If feature i was not selected by RF, then the parameters a and b are determined by adjacent values of $c_{i,j}$ in the grid partition of feature i. More specifically, $a = c_{i,j} - |c_{i,j} - c_{i,j+1}|$ and $b = c_{i,j} + |c_{i,j} - c_{i,j-1}|$.

TABLE I
EXAMPLE OF FEATURE TRANSFORMATION

original features			transformed features					
x_1	x_2	x_3	$x'_{1,1}$	$x'_{1,2}$	$x'_{2,1}$	$x'_{2,2}$	$x'_{2,3}$	x'_3
0	1	1	0.000	0.000	1	0	0	1
2	1	1	0.000	0.000	1	0	0	1
5	1	0	0.281	0.000	1	0	0	0
8	2	0	0.875	0.180	0	1	0	0
8	3	1	0.875	0.180	0	0	1	1
10	2	1	1.000	0.500	0	1	0	1
15	1	0	1.000	1.000	1	0	0	0
20	3	1	1.000	1.000	0	0	1	1

Given these membership functions we can pre-process our initial dataset of raw features in order to obtain the membership values of the $\sum_{i=1}^{m} k_i$ fuzzy sets. These membership values are subsequently used as inputs in our optimization model.

Example 1: In Table I we give an example that illustrates the feature transformation. Table I shows eight rows of a dataset that contains three raw features x_1, x_2 and x_3. In the example, x_1 is a continuous feature, x_2 is a categorical feature and x_3 is a binary feature. Suppose that after using random forest, feature x_1 is selected twice with threshold values $c_{1,1} = 7$ and $c_{1,2} = 10$. If we use an S-shaped membership function, we get that $a = \text{median}\{0, 2, 5\} = 2$ and $b = \text{median}\{8, 8, 10, 15, 20\} = 10$ for $\mu_{1,1}^{S}(x)$. For $\mu_{1,2}^{S}(x)$ we have $a = \text{median}\{0, 2, 5, 8, 8\} = 5$ and $b = \text{median}\{10, 15, 20\} = 15$. If we now apply the S-shaped membership function with these parameters on the feature x_1, we end up with the transformed features $x'_{1,1}$ and $x'_{1,2}$. Feature x_2 is transformed into features $x'_{2,1}$, $x'_{2,2}$ and $x'_{2,3}$ using one-hot encoding. Finally, feature x_3 is the same as feature x'_3 since it is binary. Fig. 2 shows a plot of the resulting membership functions for feature x_1. $\square$

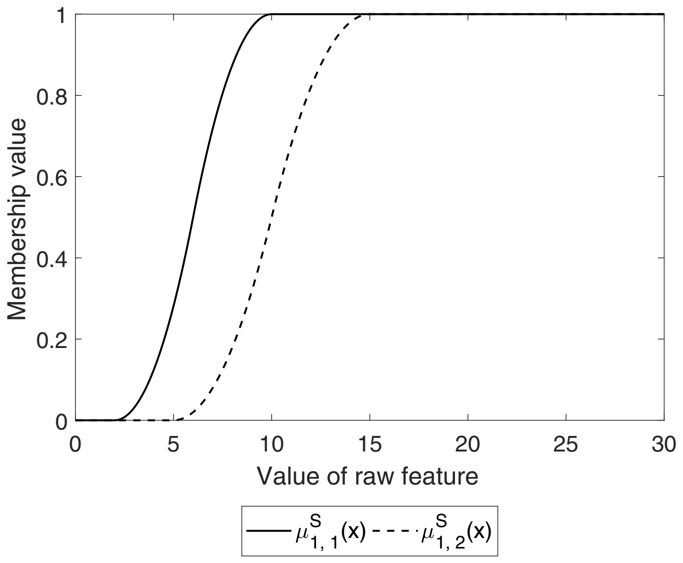

Fig. 2. Plot of membership functions from the example in Table I.

B. Fuzzy Decision trees: MIP formulation

Consider a fully grown binary tree of depth k and let the root node be denoted by u_0. The set of all nodes is given by $\mathcal{N}$. The set of all leaf nodes is given by $\mathcal{N}^L$. Define the set of all

TABLE II
SUMMARY OF NOTATION, CONSTANTS, AND VARIABLES USED IN MIP MODEL.

Symbol	Type	Definition
n	constant	number of rows in data file
m	constant	number of features in data
v	constant	number of leaves in tree
k	constant	depth of tree
$v(r,i)$	constant	feature value for data row r and feature i
$t(r)$	constant	class or label of data row r
j^-	constant	predecessor of internal node j
$\mathcal{N}^L$	set	set of leaf nodes in tree
$\mathcal{N}^B$	set	set of branching nodes in tree
$\mathcal{N}^I$	set	set of internal nodes in tree
$\mathcal{P}_l$	set	set of nodes (excluding the root node) on the path to leaf node l
$\mathcal{T}$	set	set of all labels in the dataset
$f_{i,j}$	binary	decision variable, feature i is used in decision rule of node j
$p_{l,t}$	binary	decision variable, classifier prediction of leaf l and target t
$x_{j,r}$	continuous	decision variable, membership value for data row r at the branch between node j and node j^-
$fv(j,r)$	continuous	decision variable, feature value for data row r at node j
$y_{l,r}$	continuous	decision variable, membership value for data row r at leaf node l
$pen_{l,r}$	continuous	decision variable, penalty for data row r at leaf node l

branching nodes as $\mathcal{N}^B = \{u \mid u \in \mathcal{N}, u \notin \mathcal{N}^L\}$. The set of internal nodes is then given by $\mathcal{N}^I = \{u \mid u \in \mathcal{N}, u \neq u_0\}$.

There are n rows in the data set and the dataset contains m features. Denote the class or label of data row r by $t(r)$. Let $\mathcal{T} = \cup_{r\in[1,n]}\ t(r)$ denote the set of all class labels in the dataset.

For a fully grown tree of depth k, the number of paths from the root node to a leaf node will be equal to the number of leaf nodes $v = |\mathcal{N}^L|$. For each $l \in \mathcal{N}^L$, let $\mathcal{P}'_l = \{u_0, u_1, \ldots, u_k\}$ denote a sequence of nodes starting at the root node u_0 and ending in a leaf node u_k at depth k in the tree. Define $\mathcal{P}_l = \{j \in \mathcal{P}'_l \mid j \neq u_0\}$ as the set of nodes in $\mathcal{P}'_l$ excluding the root node.

Every node $j \in \mathcal{N}^B$ in the tree needs a binary decision variable $f_{i,j} \in \{0,1\}$ to specify whether feature $i \in [1,m]$ is used in the decision rule on node j. Let $v(r,i)$ denote the value of feature i in data row r of the dataset, then the value of the selected feature at node j is given by

$$fv(j,r) = \sum_{1\le i\le m} f_{i,j}\cdot v(r,i) \quad \forall j \in \mathcal{N}^B, r \in [1,n] \qquad (1)$$

For each node $j \in \mathcal{N}^I$, let j^- denote the predecessor of node j. Define now $MR(j,r) = fv(j^-,r)$. One can interpret $MR(j,r) = fv(j^-,r)$ as the membership value or degree to which data row r takes a right branch at node j^- on the path to node j. The membership value or degree to which data row r takes a left branch at node j^- on the path to node j is then given by $ML(j,r) = 1 - fv(j^-,r)$.

Let $D_{j,l} = 1$ indicate that data row r takes a right branch at node j^- on the path to node j and to leaf node l. Note that $D_{j,l}$ is *not* a decision variable. Instead, $D_{j,l}$ is determined by the structure of the tree (by the depth k and the numbering of the nodes). For each path starting at the root node and ending in a leaf node $l \in \mathcal{N}^L$ we can define the degree to which data row r will end up in leaf l by following path $\mathcal{P}_l$ as:

$$y_{l,r} = \text{mean}\{D_{j,l}\cdot fv(j^-,r) + (1-D_{j,l})\cdot(1-fv(j^-,r)) \mid j \in \mathcal{P}_l\}$$

The usual approach in fuzzy modeling would be to use a T-norm (the most popular one is to take the minimum of membership values). For computational efficiency we use the mean instead of the minimum, since linearizing the minimum would introduce extra binary variables into the formulation. We can summarize the steps described above as follows:

- For each branching node we need to determine which feature we want to branch on.
- For each leaf node we need to determine the target or class label that is associated with it.
- The feature values associated with the feature chosen for branching will define the membership values between adjacent nodes.
- The membership values between adjacent nodes (that are on the path to a leaf node) will determine the membership value of a data row to a particular leaf node. More specifically, the membership value of a data row to a particular leaf node is equal to the average of the membership values between adjacent nodes (that are on the path to this particular leaf node).

Example 2: We can illustrate these concepts using the example dataset described in Table I. Suppose that $\{j_1, j_2, j_3\}$ is a sequence of nodes from the root node j_1 to a leaf node j_3. Furthermore, suppose that after visiting node j_1 a right branch is taken in order to reach node j_2 and suppose that a left branch is taken in order to reach node j_3. For this example we thus have $j_1, j_2 \in \mathcal{N}^B$ and $j_3 \in \mathcal{N}^L$. At node j_1 and j_2 in the tree, one column out of the six columns corresponding to the transformed features needs to be selected. Suppose that column 1 is selected at node j_1, then $f_{1,j_1} = 1$ and the membership value or degree to which data row 4 takes a right branch at node j_1 is given by 0.875. If column 3 is selected at node j_2, then $f_{3,j_2} = 1$ and the membership value or degree to which data row 4 takes a left branch at node j_2 is given by $1 - 0 = 1$. The membership value of data row 4 to leaf node j_3 is then given by mean$\{0.875, 1\}$. □

In order to complete the formulation of the MIP model we need to specify the objective function. For each data row r we define a penalty or loss $pen_{l,r}$ in each leaf $l \in \mathcal{N}^L$. The penalty $pen_{l,r}$ is defined as follows:

$$pen_{l,r} = \begin{cases} 0 & \text{if predicted class in leaf } l \text{ equals } t(r) \\ y_{l,r} & \text{otherwise} \end{cases}$$

The objective function that we aim to minimize, is the sum of the penalties $pen_{l,r}$ over all leaves and data rows. The complete formulation is as follows:

$$\min e = \sum_{1 \le r \le n} \sum_{1 \le l \le v} pen_{l,r} \tag{2}$$

subject to:

$$x_{j,r} = 2D_{j,l} \cdot fv(j^-, r) + (1 - fv(j^-, r) - D_{j,l}) \quad \forall j \in \mathcal{P}_l, l \in \mathcal{N}^L, r \in [1, n] \tag{3}$$

$$\sum_{1 \le i \le m} f_{i,j} = 1 \quad \forall j \in \mathcal{N}^B \tag{4}$$

$$f_{i,j} \in \{0, 1\} \quad \forall j \in \mathcal{N}^B, i \in [1, m] \tag{5}$$

$$y_{l,r} = (1/k) \cdot \sum_{j \in \mathcal{P}_l} x_{j,r} \quad \forall l \in \mathcal{N}^L, r \in [1, n] \tag{6}$$

$$\sum_{t \in \mathcal{T}} p_{l,t} = 1 \quad \forall l \in \mathcal{N}^L \tag{7}$$

$$p_{l,t} \in \{0, 1\} \quad \forall l \in \mathcal{N}^L, t \in \mathcal{T} \tag{8}$$

$$pen_{l,r} \ge y_{l,r} - p_{l,t}, \quad t = t(r) \quad \forall l \in \mathcal{N}^L, r \in [1, n] \tag{9}$$

$$0 \le pen_{l,r} \le 1 \quad \forall l \in \mathcal{N}^L, r \in [1, n] \tag{10}$$

The decision variables are $f_{i,j}, x_{j,r}, y_{l,r}, p_{l,t}, pen_{l,r}$.

Equation (3) determines the membership values along the individual branches of the tree, where $fv(j, r)$ is given by Equation (1). Equations (4) and (5) ensure that only one feature is selected to branch on at each branching node. Equation (6) determines the membership value along each path by calculating the mean of the membership values along the individual branches. Equations (7) and (8) ensure that precisely one label is predicted in each leaf node. Equation (9) determines the penalty for each leaf node. Finally, Equation (10) ensures that the penalties are bounded.

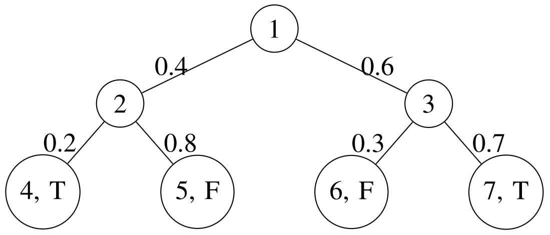

Fig. 3. Example of a learned fuzzy decision tree of depth 2. There are 3 branching nodes (node 1, 2 and 3) and 4 leaf nodes (node 4, 5, 6 and 7). The labels (F and T) in each leaf node are also displayed. The number on the branches represent membership values to fuzzy sets for a specific row of data. In particular, the value of 0.6 indicates that $\mu_{ij}(v_i) = 0.6$, where $\mu_{ij}(x)$ is the membership function of raw feature i to fuzzy set j with associated threshold of $c_{i,j}$.

C. Evaluation of learned trees

After a decision tree has been learned, the next step is the evaluation of the tree. Given a learned tree and a data row from a testing dataset, there are multiple ways to determine the final prediction. The final prediction is determined by the choice of the defuzzification method. Note that this issue is not relevant for regular or crisp decision trees, since in that case a particular row of data will end up in exactly one leaf node. In the case of our fuzzy decision tree, a row of data can end up in all leaves and each path to a leaf has a corresponding membership value (or weight).

In our experiments we used the following four methods for evaluating/defuzzifying a learned tree: (1) MEAN-MAXCLASS, (2) MEAN-MAXAVG, (3) MIN-MAXCLASS, and (4) MIN-MAXAVG. In MEAN-MAXCLASS we used the mean of the membership values along a particular path to compute the membership value of a data row to a leaf. The predicted class label is equal to the label in the leaf with the highest membership value (ties are randomly broken). In MEAN-MAXAVG we also use the mean the membership values along a particular path to compute the membership value of a data row to a leaf. But in the case of MEAN-MAXAVG the predicted class label is equal to the label with the highest average membership values over all leaves. The definitions of MIN-MAXCLASS and MIN-MAXAVG are similar to the ones given above, except that they use the minimum of the membership values along a particular path to compute the membership value of a data row to a leaf.

Example 3: We illustrate the first two defuzzification methods on a simple example using Fig. 3. If we use defuzzification method (1)-(2) we have $0.3, 0.6, 0.45, 0.65$ as the membership values (of this row of data) to the four leaves. The prediction using MEAN-MAXCLASS would be class T since that is the label in the leaf with the highest membership value.
Using MEAN-MAXAVG the prediction would be class F since the mean of 0.6 and 0.45 is higher than the mean of 0.3 and 0.65. □

III. Experiments

A. Setup of Experiments

We conducted tests on a number of benchmark datasets that have been taken from the UCI machine learning repository [21]. We compared the performance of the classification method from Scikit-learn [22], a machine learning package in Python, and our fuzzy decision tree method that is solved using CPLEX [23]. We set the time limit for solving problems in CPLEX to 30 minutes. We learn decision trees with depths ranging from 1 to 4. We also compared the performance of our method with the performance of a FIS estimated using MATLAB.

We used the following datasets: Iris, Indian Liver Patients, Blood Transfusion, Prima Indian Diabetes, Wine and Seeds. These datasets have also been used in previous studies on decision trees such as [7], [12] and thus serve as a useful benchmark. We took 5 independent splits of the data into training (70% of the data) and testing (30% of the data). For each split, each training dataset was pre-processed according to the steps detailed in Section II in order to derive the membership functions and to learn the fuzzy decision tree. These same membership functions were then applied on raw features in the testing data and fed to the learned tree in order to determine predictions for the testing data. The reported accuracy is averaged over these 5 splits/samples of training and testing data.

As the FIS in MATLAB can only deal with binary classification problems where all the features are continuous, we can only compare the performance of our method relative to the performance of FIS on the Blood transfusion and Diabetes datasets. We estimated a first-order Sugeno-type FIS with 5 and 10 rules. The input membership function type was gaussian and the rule-base was initialized with fuzzy c-means clustering using the MATLAB function `genfis3`. The output membership function type is linear. The FIS was subsequently tuned using the MATLAB function `anfis`. Since the output from the FIS is a number in the range $[0, 1]$, we used a threshold value 0.5 in order assign a class label to a row of data. The results using another number of rules in the range 3-10 are similar and are not reported in order to save space. The results for FDTIP are based on a random forest with $N_T = 100$ trees and with $N_G = 20$. We display results for the S-shaped membership function as the results using the Z-shaped membership function are very similar.

B. Results

The results from our experiments are summarized in Table III and Table IV. Table III shows the average classification accuracy on the training data. The optimality gap, as reported by CPLEX, of the corresponding MIP is also displayed.

On the datasets where it is possible to use a FIS, we see that FIS performs quite well compared to CART and FDTIP. FDTIP tends to have a lower accuracy on training data compared to CART and FIS. One possible explanation for this could be the specification of the objective function in FDTIP.

TABLE III
Average classification accuracy and optimality gap on training datasets for trees with depth 1 to 4.

		d = 1	d=2	d=3	d= 4
Iris	MEAN-MAXCLASS	0.691	0.958	0.943	0.947
	MIN-MAXCLASS	0.691	0.958	0.943	0.950
	MEAN-MAXAVG	0.691	0.958	0.930	0.954
	MIN-MAXAVG	0.691	0.958	0.907	0.926
	CART	0.691	0.966	0.981	0.992
	FDTIP (OPT Gap)	0.000	0.000	0.000	0.189
Diabetes	MEAN-MAXCLASS	0.731	0.735	0.734	0.642
	MIN-MAXCLASS	0.731	0.739	0.738	0.642
	MEAN-MAXAVG	0.731	0.738	0.738	0.651
	MIN-MAXAVG	0.731	0.735	0.730	0.647
	CART	0.733	0.767	0.771	0.811
	FDTIP (OPT Gap)	0.000	0.000	0.015	0.894
	FIS 5 rules	0.803	0.803	0.803	0.803
	FIS 10 rules	0.843	0.843	0.843	0.843
Transfusion	MEAN-MAXCLASS	0.774	0.771	0.771	0.771
	MIN-MAXCLASS	0.774	0.769	0.771	0.771
	MEAN-MAXAVG	0.774	0.769	0.771	0.771
	MIN-MAXAVG	0.774	0.771	0.770	0.765
	CART	0.769	0.769	0.803	0.813
	FDTIP (OPT Gap)	0.000	0.000	0.000	0.000
	FIS 5 rules	0.805	0.805	0.805	0.805
	FIS 10 rules	0.814	0.814	0.814	0.814
Liver	MEAN-MAXCLASS	0.716	0.712	0.712	0.712
	MIN-MAXCLASS	0.716	0.712	0.712	0.712
	MEAN-MAXAVG	0.716	0.712	0.712	0.712
	MIN-MAXAVG	0.716	0.712	0.712	0.712
	CART	0.712	0.713	0.735	0.762
	FDTIP (OPT Gap)	0.000	0.000	0.032	0.884
Seeds	MEAN-MAXCLASS	0.684	0.852	0.805	0.839
	MIN-MAXCLASS	0.684	0.852	0.805	0.842
	MEAN-MAXAVG	0.684	0.872	0.804	0.876
	MIN-MAXAVG	0.684	0.873	0.799	0.853
	CART	0.669	0.905	0.946	0.978
	FDTIP (OPT Gap)	0.000	0.000	0.000	0.297
Wine	MEAN-MAXCLASS	0.696	0.875	0.858	0.901
	MIN-MAXCLASS	0.696	0.877	0.859	0.880
	MEAN-MAXAVG	0.698	0.896	0.856	0.920
	MIN-MAXAVG	0.698	0.899	0.862	0.912
	CART	0.691	0.936	0.992	0.997
	FDTIP (OPT Gap)	0.000	0.000	0.000	0.391

Unlike CART, FDTIP does not use the exact labels of the data rows to partition the data, but instead uses a surrogate penalty function to count mis-classifications. The fact that the objective function of FDTIP does not count the exact mis-classifications during training, may explain the pattern observed in Table III. The good performance of FIS might be due to the fact that the parameters of the membership functions used in FIS can be tuned during training. Another reason could be related to the fact that the output variable is continuous (a probability) so that the FIS can use interpolation to improve the accuracy.

From an optimization perspective, we see that the optimality gap of the associated MIP for trees up to depth 3 is zero in most cases, indicating that the problem has been solved to optimality. For depth 4, we see that CPLEX has a much harder problem to solve within the time limit of 30 minutes and this is also reflected in the larger optimality gap.

Table IV shows the average classification accuracy on the testing data. On the Iris, Diabetes, Transfusion and Liver datasets, the three methods are relatively close to each other (at

TABLE IV
AVERAGE CLASSIFICATION ACCURACY ON TESTING DATASETS FOR TREES WITH DEPTH 1 TO 4.

		d = 1	d=2	d=3	d= 4
Iris	MEAN-MAXCLASS	0.609	0.947	0.960	0.933
	MIN-MAXCLASS	0.609	0.947	0.956	0.947
	MEAN-MAXAVG	0.609	0.947	0.960	0.951
	MIN-MAXAVG	0.609	0.947	0.933	0.924
	CART	0.609	0.924	0.956	0.956
Diabetes	MEAN-MAXCLASS	0.763	0.767	0.765	0.674
	MIN-MAXCLASS	0.763	0.763	0.762	0.674
	MEAN-MAXAVG	0.763	0.762	0.762	0.684
	MIN-MAXAVG	0.763	0.752	0.751	0.675
	CART	0.742	0.766	0.764	0.734
	FIS 5 rules	0.767	0.767	0.767	0.767
	FIS 10 rules	0.753	0.753	0.753	0.753
Transfusion	MEAN-MAXCLASS	0.752	0.746	0.746	0.746
	MIN-MAXCLASS	0.752	0.746	0.746	0.746
	MEAN-MAXAVG	0.752	0.746	0.746	0.746
	MIN-MAXAVG	0.752	0.746	0.746	0.746
	CART	0.746	0.746	0.766	0.769
	FIS 5 rules	0.770	0.770	0.770	0.770
	FIS 10 rules	0.768	0.768	0.768	0.768
Liver	MEAN-MAXCLASS	0.723	0.721	0.721	0.721
	MIN-MAXCLASS	0.723	0.721	0.721	0.721
	MEAN-MAXAVG	0.723	0.721	0.721	0.721
	MIN-MAXAVG	0.721	0.721	0.721	0.721
	CART	0.721	0.719	0.676	0.672
Seeds	MEAN-MAXCLASS	0.625	0.838	0.762	0.787
	MIN-MAXCLASS	0.625	0.841	0.771	0.787
	MEAN-MAXAVG	0.625	0.857	0.749	0.889
	MIN-MAXAVG	0.625	0.854	0.752	0.857
	CART	0.590	0.848	0.886	0.889
Wine	MEAN-MAXCLASS	0.619	0.830	0.830	0.853
	MIN-MAXCLASS	0.619	0.834	0.842	0.823
	MEAN-MAXAVG	0.619	0.864	0.849	0.902
	MIN-MAXAVG	0.619	0.857	0.857	0.879
	CART	0.608	0.864	0.936	0.902

depth 1-3) in terms of accuracy as the difference in accuracy is at most 3-4% in almost all cases. For Iris, Diabetes and Liver datasets, the performance of FDTIP is at least as good as CART and FIS for most of the defuzzification methods. FDTIP is even able to outperform CART and FIS for some of the defuzzification methods but the differences are not that large. On the Seeds and Wine datasets we see a different pattern. For depth 1,2 and 4 the performance of FDTIP is very similar to CART, but not at depth 3. The results in Table IV can also be used to determine which defuzzification methods perform the best. Overall, the results show that the MIN-MAXAVG and MEAN-MAXAVG defuzzification methods tend to perform the best across all datasets.

It is interesting to compare the performance of FDTIP on training data versus testing data. In general, the accuracy on training data and testing data is much closer to each other for FDTIP compared to CART. Another observation is that, although FDTIP did not have the best performance on training data, it is still competitive with CART and FIS on the testing data. This shows that fuzzifying the decision boundaries of a crisp decision tree has added value in terms of classification accuracy on testing data. This also suggests that there might be some inherent trade-off associated with the use of fuzzy decision trees: the ability to generalize well on testing data, comes at the cost of the performance on training data.

IV. CONCLUSION

We have proposed a new modeling framework for learning fuzzy decision trees that uses Mixed Integer Linear Programing to solve the associated optimization problem. The initial experimental results are promising and show that our method has a performance similar to both CART and FIS in most cases and is able to outperform in them some cases as well. Our framework is complementary to FIS and offers an alternative way for fuzzy modeling using decision trees that suitable for situations in which a FIS cannot be applied.

The research presented in this paper is a first step to designing a general framework for learning fuzzy decision trees. As future work, we will study alternative loss functions and defuzzification methods in order to further improve the performance of our method. Another interesting research direction is to extend the current framework such that the parameters of the membership functions are estimated using data during the optimization process. Moreover, we will investigate how to combine learned fuzzy decision trees with optimization problems, like in [24]. We expect that predictions drawn from fuzzy decision trees are less sensitive to prediction errors, hence, the optimization method that takes such predictions as parameters will be more robust.

In our analysis we have not explicitly considered the complexity and interpretability of the rules that are obtained by our method. Rulebase simplification [25] has been proposed to deal with cases where the obtained fuzzy sets overlap too much and cannot be interpreted linguistically. In future work we will investigate the applicability of such methods to our problem and we will investigate whether the interpretability/accuracy trade-off can be incorporated in the optimization process.

ACKNOWLEDGMENT

This work was partially supported by NWO [project number 628.010.001] and EU EUROSTARS [Project E! 11582].

REFERENCES

[1] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and regression trees*. Wadsworth International Group, 1984.

[2] L. Hyafil and R. L. Rivest, "Constructing optimal binary decision trees is np-complete," *Information Processing Letters*, vol. 5, no. 1, pp. 15 – 17, 1976. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0020019076900958

[3] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[4] A. Altay and D. Cinar, *Fuzzy Decision Trees*. Cham: Springer International Publishing, 2016, pp. 221–261. [Online]. Available: https://doi.org/10.1007/978-3-319-39014-7_13

[5] T. Wang, Z. Li, Y. Yan, and H. Chen, "A survey of fuzzy decision tree classifier methodology," in *Fuzzy Information and Engineering: Proceedings of the Second International Conference of Fuzzy Information and Engineering (ICFIE)*, B.-Y. Cao, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 959–968. [Online]. Available: https://doi.org/10.1007/978-3-540-71441-5_104

[6] C. Z. Janikow, "Fuzzy decision trees: issues and methods," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 28, no. 1, pp. 1–14, Feb 1998.

[7] B. Chandra and P. P. Varghese, "Fuzzifying gini index based decision trees," *Expert Systems with Applications*, vol. 36, no. 4, pp. 8549 – 8559, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417408007537

[8] K. P. Bennett and J. A. Blue, "Optimal decision trees," R.P.I. Math Report No. 214, Rensselaer Polytechnic Institute, Tech. Rep., 1996.

[9] M. Norouzi, M. D. Collins, M. Johnson, D. J. Fleet, and P. Kohli, "Efficient non-greedy optimization of decision trees," in *Proceedings of the 28th International Conference on Neural Information Processing Systems*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, pp. 1729–1737. [Online]. Available: http://dl.acm.org/citation.cfm?id=2969239.2969432

[10] D. Bertsimas and R. Shioda, "Classification and regression via integer optimization," *Operations Research*, vol. 55, no. 2, pp. 252–271, 2007.

[11] D. Bertsimas and J. Dunn, "Optimal classification trees," *Machine Learning*, vol. 106, no. 7, pp. 1039–1082, Jul 2017. [Online]. Available: https://doi.org/10.1007/s10994-017-5633-9

[12] S. Verwer and Y. Zhang, "Learning decision trees with flexible constraints and objectives using integer optimization," in *Integration of AI and OR Techniques in Constraint Programming: 14th International Conference, CPAIOR 2017, Padua, Italy, June 5-8, 2017, Proceedings*, D. Salvagnin and M. Lombardi, Eds. Cham: Springer International Publishing, 2017, pp. 94–103. [Online]. Available: https://doi.org/10.1007/978-3-319-59776-8_8

[13] Y. Yuan and M. J. Shaw, "Induction of fuzzy decision trees," *Fuzzy Sets and Systems*, vol. 69, no. 2, pp. 125 – 139, 1995. [Online]. Available: http://www.sciencedirect.com/science/article/pii/016501149400229Z

[14] X. Wang, B. Chen, G. Qian, and F. Ye, "On the optimization of fuzzy decision trees," *Fuzzy Sets and Systems*, vol. 112, no. 1, pp. 117 – 125, 2000. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0165011497003862

[15] C. Olaru and L. Wehenkel, "A complete fuzzy decision tree technique," *Fuzzy Sets Syst.*, vol. 138, no. 2, pp. 221–254, Sep. 2003. [Online]. Available: http://dx.doi.org/10.1016/S0165-0114(03)00089-7

[16] P. Flach, *Machine learning: the art and science of algorithms that make sense of data*. Cambridge University Press, 2012.

[17] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324

[18] J. S. R. Jang, "Structure determination in fuzzy modeling: a fuzzy cart approach," in *Proceedings of 1994 IEEE 3rd International Fuzzy Systems Conference*, Jun 1994, pp. 480–485 vol.1.

[19] T. Tani, M. Sakoda, and K. Tanaka, "Fuzzy modeling by id3 algorithm and its application to prediction of heater outlet temperature," in *[1992 Proceedings] IEEE International Conference on Fuzzy Systems*, Mar 1992, pp. 923–930.

[20] Z. Chi and H. Yan, "Id3-derived fuzzy rules and optimized defuzzification for handwritten numeral recognition," *IEEE Transactions on Fuzzy Systems*, vol. 4, no. 1, pp. 24–31, Feb 1996.

[21] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[22] scikit-learn: machine learning in Python. [Online]. Available: http://scikit-learn.org/

[23] IBM ILOG CPLEX Optimizer. [Online]. Available: http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/

[24] S. Verwer, Y. Zhang, and Q. C. Ye, "Auction optimization using regression trees and linear models as integer programs," *Artificial Intelligence*, vol. 244, pp. 368–395, 2017.

[25] M. Setnes, R. Babuska, U. Kaymak, and H. R. van Nauta Lemke, "Similarity measures in fuzzy rule base simplification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 28, no. 3, pp. 376–386, Jun 1998.