Look-Ahead Based Fuzzy Decision Tree Induction

Ming Dong, *Student Member, IEEE,* and Ravi Kothari, *Senior Member, IEEE*

***Abstract*—Decision tree induction is typically based on a top-down greedy algorithm that makes locally optimal decisions at each node. Due to the greedy and local nature of the decisions made at each node, there is considerable possibility of instances at the node being split along branches such that instances along some or all of the branches require a large number of additional nodes for classification. In this paper, we present a computationally efficient way of incorporating look-ahead into fuzzy decision tree induction. Our algorithm is based on establishing the decision at each internal node by jointly optimizing the node splitting criterion (information gain or gain ratio) and the classifiability of instances along each branch of the node. Simulations results confirm that the use of the proposed look-ahead method leads to smaller decision trees and as a consequence better test performance.**

***Index Terms*—Decision tree, classification, fuzzy ID3, fuzzy systems, gain ratio.**

I. Introduction

DECISION trees represent a simple and powerful method of induction from labeled instances [1], [2]. At a given node of the tree, a *stopping criterion* (based on the fraction of instances correctly classified or the cardinality of the compatible instances or some other similar measure) is used to determine if additional child nodes are necessary. If additional child nodes are required, they are added based on a *node splitting criterion* and the process is repeated for each of the new internal nodes until a completely discriminating tree is obtained. This standard approach to decision tree construction thus, corresponds to a top-down greedy algorithm that makes locally optimal decisions at each node.

One of the strengths of decision trees compared to other methods of induction is the ease with which they can be extended to nonnumeric domains. This allows decision trees to be used in situations where considerable cognitive uncertainty is present and the representation of the instances is in terms of symbolic or fuzzy attributes [3]–[8]. The uncertainty may be present in obtaining numeric values of the attributes or in obtaining the exact class to which a specific instance belongs. Fuzzy decision trees have also been used either in conjunction with or as precursors to other empirical model construction paradigms [9]–[11]. Of course, there exists close parallels between the development in the areas of crisp and fuzzy decision tree induction.

One of the challenges in decision tree induction is to develop algorithms that produce decision trees of small size and depth. In part, smaller decision trees lead to lesser computational expense in determining the class of a test instance. More significantly, however, larger decision trees lead to poorer generalization (test) performance [12]. Motivated by these considerations, a large number of algorithms have been proposed toward producing smaller decision trees. Broadly these may be classified into three categories.

1) The first category includes those efforts that are based on different criteria to split the instances at each node. Some examples of the different node splitting criteria include entropy or its variants [1], the chi-square statistic [13], [14], the G statistic [14], and the GINI index of diversity [12]. Despite these efforts, there appears to be no single node splitting that performs the best in all cases [15], [16]; nonetheless there is little doubt that random splitting performs the worst.
2) The second category is based on pruning a decision tree either during the construction of the tree or after the tree has been constructed. In either case, the idea is to remove branches will little statistical validity (see for example, [17], [18]).
3) The third category of efforts toward producing smaller decision trees is motivated by the fact that a locally optimum decision at a node may give rise to the possibility of instances at the node being split along branches, such that instances along some or all of the branches require a large number of additional nodes for classification. The so called *look-ahead methods* attempt to establish a decision at a node by analyzing the classifiability of instances along each of the branches of a split [19]–[21]. Surprisingly, mixed results (ranging from look-ahead *makes no difference* to look-ahead *produces larger trees* [21]) are reported in the literature when look-ahead is used.

Of these approaches, look-ahead has been the least frequently studied. In addition, the fact that it produces mixed results is counter intuitive. In this paper, we propose a novel method of evaluating the classifiability of instances along the branches of a node split [22]. Based on this method, we propose a fuzzy decision tree induction algorithm that utilizes look-ahead to produce smaller decision trees. We have organized the rest of the paper as follows. In Section II, we briefly review Fuzzy ID3 (F-ID3) [5] as well as the gain ratio (GR) method [2] and in that context, establish the need for examining the structure of the instances in each branch of a node. In Section III, we propose a new mechanism for look-ahead and propose a look-ahead based fuzzy decision tree induction algorithm. Experimental results and presented in Section IV and our conclusions appear in Section V.

Manuscript received November 17, 2000; revised February 26, 2001.

The authors are with the Artificial Neural Systems Laboratory, Department of Electrical and Computer Engineering and Computer Science, University of Cincinnati, Cincinnati, OH 45221-0030 USA.

Publisher Item Identifier S 1063-6706(01)04534-9.

II. Fuzzy Decision Tree Induction

In this section, we introduce the notation used in the rest of the paper and briefly examine the popular F-ID3 algorithm [5]

TABLE I
SYNTHETIC DATA TO MOTIVATE THE NEED FOR LOOK-AHEAD. NUMBERS IN THE TABLE ARE THE MEMBERSHIP VALUES

Outlook			Temperature			Humidity		Wind		Plan		
Sunny	Cloudy	Rain	Hot	Mild	Cool	Humid	Dry	Windy	Calm	A	B	C
0.7	0.1	0.0	1.0	0.0	0.0	0.8	0.2	0.4	0.6	0.0	0.4	0.6
0.3	0.8	0.0	0.6	0.4	0.0	0.0	1.0	0.0	1.0	1.0	0.3	0.0
0.0	0.7	0.3	0.8	0.2	0.0	0.1	0.9	0.2	0.8	0.3	0.6	0.1
0.2	0.7	0.1	0.3	0.7	0.0	0.2	0.8	0.3	0.7	0.9	0.1	0.0
0.0	0.1	0.9	0.7	0.3	0.0	0.5	0.5	0.5	0.5	0.0	0.0	1.0
0.0	0.7	0.3	0.0	0.3	0.7	0.7	0.3	0.4	0.6	0.2	0.0	0.8
0.0	0.3	0.7	0.0	0.0	1.0	0.0	1.0	0.1	0.9	0.0	0.0	1.0
0.0	1.0	0.0	0.0	0.2	0.8	0.2	0.8	0.0	1.0	0.7	0.0	0.3
0.5	0.7	0.0	1.0	0.0	0.0	0.6	0.4	0.7	0.3	0.2	0.8	0.0
0.5	0.6	0.0	0.0	0.3	0.7	0.0	1.0	0.9	0.1	0.0	0.3	0.7
0.7	0.3	0.0	1.0	0.0	0.0	1.0	0.0	0.2	0.8	0.4	0.9	0.0
0.2	0.6	0.2	0.0	1.0	0.0	0.3	0.7	0.3	0.7	0.7	0.2	0.1
0.9	0.1	0.0	0.2	0.8	0.0	0.1	0.9	1.0	0.0	0.0	0.0	1.0
0.0	0.9	0.1	0.0	0.9	0.1	0.1	0.9	0.7	0.3	0.0	0.0	1.0
0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.8	0.2	0.0	0.0	1.0
1.0	0.0	0.0	0.8	0.5	0.0	0.0	1.0	0.0	1.0	0.4	0.5	0.0

as well as the GR [2] method of inducing decision trees. Though we integrate the proposed look-ahead method with F-ID3 and GR, it is in fact easy to integrate it with other node splitting criterion, for example, the one based on minimum classification ambiguity [4].

Let the universe of objects be described by n attributes $A = \{A^{(1)}, \ldots, A^{(n)}\}$. An attribute $A^{(k)}$ takes m_k values of fuzzy subsets $A_1^{(k)}, A_2^{(k)}, \ldots, A_{m_k}^{(k)}$. There are a total of N training instances. Based on the attributes, an object is classified into C fuzzy subsets $\omega_1, \omega_2, \ldots, \omega_C$.

F-ID3 is based on an adaptation of the classical definition of entropy. This adaptation, or fuzzy entropy as it is also called, for a subset can be defined as

$$H_i^{(k)} = \sum_{j=1}^{C} -p_i^{(k)}(j) \log p_i^{(k)}(j) \tag{1}$$

where, $p_i^{(k)}(j)$ is the relative frequency of the ith subset of attribute k with respect to ω_j $(1 \leq j \leq C)$ and defined as

$$p_i^{(k)}(j) = \frac{M\left(A_i^{(k)} \cap \omega_j\right)}{M\left(A_i^{(k)}\right)} \tag{2}$$

and $M(\cdot)$ is the cardinality of a fuzzy set [4], [18]. The attribute h chosen to split the instances at a given node is based o,

$$h = \arg \min_{1 \leq k \leq n} E^{(k)} \tag{3}$$

where, arg returns that value of the index (k) for which $E^{(k)}$ is the smallest and

$$E^{(k)} = \sum_{i=1}^{m_k} \frac{M\left(A_i^{(k)}\right)}{\sum_{j=1}^{m_k} M\left(A_j^{(k)}\right)} H_i^{(k)}. \tag{4}$$

It is easy to see that choosing an attribute to split the instances based on the above equations results in the maximum decrease in fuzzy entropy, i.e., one that maximizes the cardinality of compatible instances.

F-ID3, however, has an obvious drawback just like ID3 algorithm. It favors attributes with a large number of possible attribute values. To discourage a large number of partitions, a factor based on the entropy of the size of the splits was proposed giving rise to the GR measure [2]. More specifically

$$\mathrm{GR}^{(k)} = \mathrm{IG}^{(k)} / \mathrm{IV}^{(k)} \tag{5}$$

where $\mathrm{IG}^{(k)}$ is the information gain defined as $E - E^{(k)}$ with E being the fuzzy entropy at a node, $E^{(k)}$ is as given by (4) and $\mathrm{IV}^{(k)}$ is the information value of attribute $A^{(k)}$ defined as

$$\mathrm{IV}^{(k)} = \sum_{i=1}^{m_k} - \left(\frac{M\left(A_i^{(k)}\right)}{M\left(A^{(k)}\right)}\right) \log \left(\frac{M\left(A_i^{(k)}\right)}{M\left(A^{(k)}\right)}\right). \tag{6}$$

One problem is that GR might choose attributes with very low IV scores, rather than those with high information gain. To avoid this, we can simply calculate the average fuzzy entropy value for all the attributes at a node then select only from among those with below average fuzzy entropy values.

Irrespective of whether information gain or the gain ratio is used, the construction of a decision tree represents a greedy search strategy that implements a locally optimum decision at each node. The difficulty is that a combination of locally optimal decisions can not guarantee the global optimum tree i.e., a tree with smallest size. Indeed it is an NP-hard problem to find the smallest tree or one with the least number of leaves [18].

To illustrate this further and to motivate the consideration of look-ahead in the construction of a decision tree we present an illustrative example. Consider the data in Table I, which is a modification of the data used in [4]. There are four attributes.

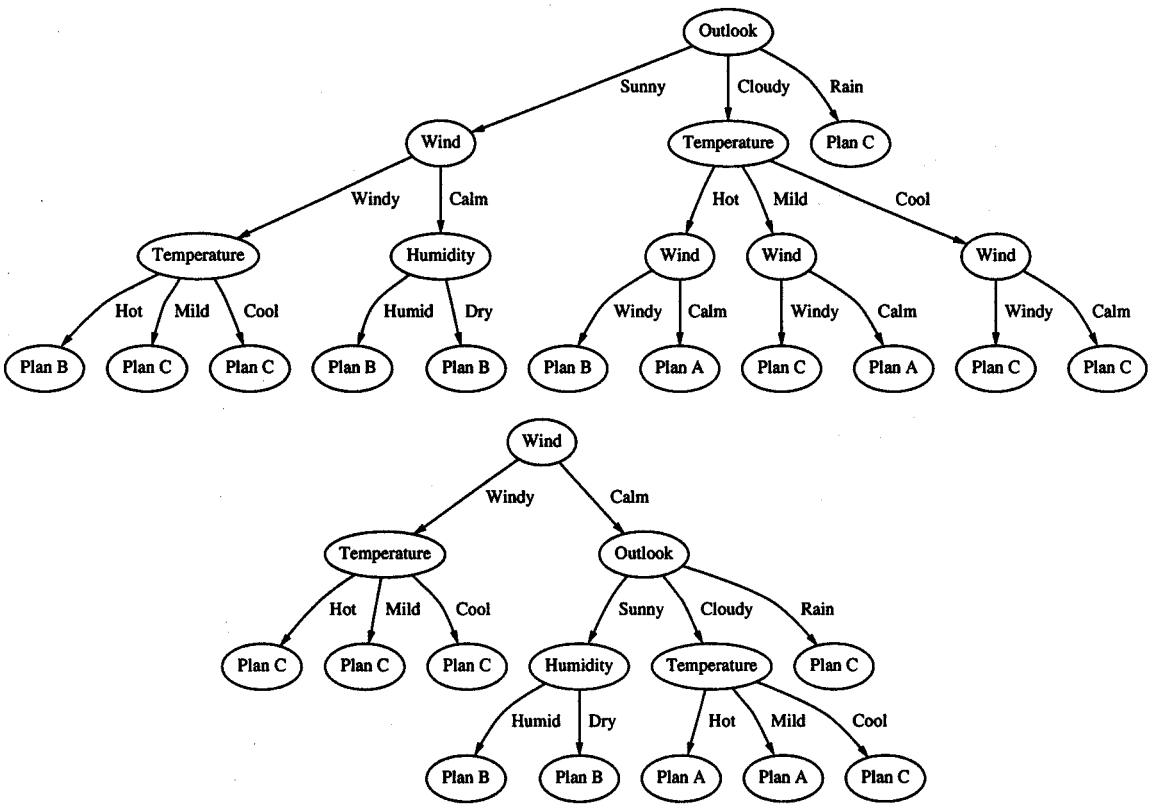

Fig. 1. The decision trees generated by choosing the attribute with the lowest fuzzy entropy at each node (top). The bottom panel shows an alternative and smaller decision tree that is possible. In these trees, a node is considered terminal if the cardinality $M(\cdot)$ is less than one.

These attributes and the values they can assume are defined below

$$\begin{aligned} \mathrm{A} &= \{\mathrm{Outlook},\ \mathrm{Temperature},\ \mathrm{Humidity},\ \mathrm{Wind}\} \\ \mathrm{Outlook} &= \{\mathrm{Sunny},\ \mathrm{Cloudy},\ \mathrm{Rain}\} \\ \mathrm{Temperature} &= \{\mathrm{Hot},\ \mathrm{Mild},\ \mathrm{Cool}\} \\ \mathrm{Humidity} &= \{\mathrm{Humid},\ \mathrm{Dry}\} \\ \mathrm{Wind} &= \{\mathrm{Windy},\ \mathrm{Calm}\}. \end{aligned}$$

There are three classes with class labels `Plan A`, `Plan B`, and `Plan C`. The numbers in Table I are the membership values. Membership values are not probabilities, thus, it is not necessary for membership values of all linguistic terms of an attribute to add to one.

The construction of the tree, based on F-ID3 proceeds as follows. At the root node, the fuzzy entropy for all the attributes based on (4) are

$$\begin{aligned} E(\mathrm{Outlook}) &= E^{(1)} = 0.8613 \\ E(\mathrm{Temperature}) &= E^{(2)} = 0.9297 \\ E(\mathrm{Humidity}) &= E^{(3)} = 1.0291 \\ E(\mathrm{Wind}) &= E^{(4)} = 0.9366 \end{aligned}$$

Thus, using (3), we choose the attribute `Outlook`, which has the lowest fuzzy entropy at the root node. Repeating this further for each of the child nodes, we arrive at the decision tree shown in the top panel of Fig. 1. On the contrary, if attribute `Wind` (which does not have the lowest fuzzy entropy but leads to a simpler distribution of patterns at the child nodes) is chosen, a smaller sized tree is obtained as shown in the bottom panel of Fig. 1. In these trees, a node is considered terminal if the cardinality $M(\cdot)$ of incorrectly classified instances is less than one. It is because of this that one finds all child nodes with the same class label. This implies that $M(\cdot)$ at the parent node was greater than 1; however at each the child nodes $M(\cdot)$ is less than 1. This example, though simple, illustrates that choosing an attribute based on jointly considering the node splitting criterion (information gain here) and classifiability of the instances along all the child branches can lead to smaller decision trees[1].

A similar situation arises when GR is considered as shown below

$$\begin{aligned} \mathrm{GR}(\mathrm{Outlook}) &= \mathrm{GR}^{(1)} = 0.1953 \\ \mathrm{GR}(\mathrm{Temperature}) &= \mathrm{GR}^{(2)} = 0.1226 \\ \mathrm{GR}(\mathrm{Humidity}) &= \mathrm{GR}^{(3)} = 0 \\ \mathrm{GR}(\mathrm{Wind}) &= \mathrm{GR}^{(4)} = 0.1943. \end{aligned}$$

As before we would still choose `Outlook` which has the highest GR at the root node. In this case, the overall tree has 19 nodes (not shown), which is still larger than that shown in the bottom panel of Fig. 1.

III. Lookahead Based Fuzzy Decision Tree Induction

The goal of look-ahead is to evaluate the classifiability of instances that are split along branches of a give node. *The question*

[1]This smaller tree was actually obtained using the algorithm we propose in this paper. Details of this algorithm appear in Section III.

is how to evaluate the classifiability? Prior efforts (within the context of crisp decision trees) used a classifier parameterized identically to the one producing the split to evaluate the classifiability of the instances along each branch of the split. For example, when a linear discriminant is used to split the instances, then a linear discriminant is used to evaluate the instances in each of the two subsets of instances resulting from the split. This corresponds to a single step look-ahead. The difficulty however, is that to obtain a good estimate of the classifiability one would typically need a multilevel look-ahead. Even so, it is possible that in a situation where three–step look-ahead is used, a very different outcome would result if a four-step look-ahead is used. The argument can be generalized and it becomes clear that if a classifier is actually used to evaluate the classifiability then an exhaustive look-ahead is necessary. Clearly, this is not feasible in all but the most trivial of cases. As earlier efforts used a few (one or two mostly) step look-ahead, they did not consistently obtain better results [19]–[21]. Indeed, in some cases they obtained poorer results with look-ahead than without.

We propose to characterize the classifiability of instances that are split along branches of a give node using a nonparametric method. The motivation for our approach can be more easily explained within the context of a crisp dataset. Consider a two-class situation with n attributes. In $(n+1)$ dimensions (n variables and one for the class label), one may thus, visualize a surface. When the instances of a class are interlaced with another class, this surface is rough. However, when there are compact and disjoint class regions, this surface is considerably smoother. This is akin to visualizing the class label as defining the surface in n dimensions. Fig. 2 shows the distribution of some instances in two-dimensions (2-D). The third dimension is the class label (zero for one class and one for the other). It is clear that the distribution in the bottom panel is more amenable to subsequent classification. It may also be observed that the surface formed by the class label is considerably smoother in the bottom panel. Thus, we propose to evaluate the classifiability of instances that are split along branches of a give node in terms of the smoothness of the class label surface of instances assigned to that branch. A widely used way of characterizing the smoothness (or roughness) of a surface in image processing is through the use of a cooccurrence matrix [23] and we adopt it here with some modifications to characterize the texture of the class-label surface[2].

In the following, we formally define the characterization of the class-label surface.

Definition 1: Let $\mu_i^{(k)}(x)$ $(1 \leq k \leq n, 1 \leq i \leq m_k, 1 \leq x \leq N)$ denote the membership value of instance x for the ith value of kth attribute. The distance between instance x and y is defined by

$$D_{xy} = \sum_{k=1}^{n} \sum_{i=1}^{m_k} \left| \mu_i^{(k)}(x) - \mu_i^{(k)}(y) \right|. \tag{7}$$

For any instance x in the data set, we can find those instances that are within a circular neighborhood of radius r, of instance x, based on the distance defined above. We can then define the local cooccurrence matrix P for instance x as follows.

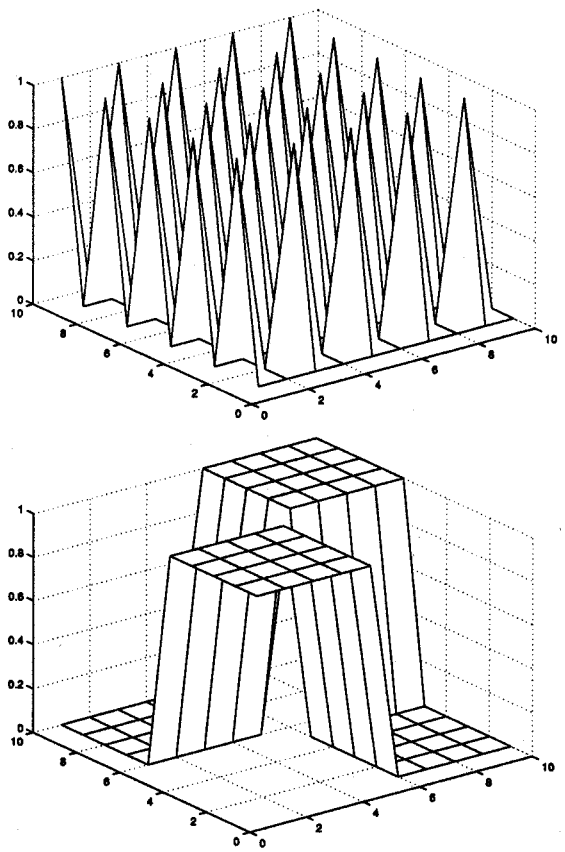

Fig. 2. Three-dimensional plot to show the smoothness of the surface caused by different distribution of instances. The distribution in the top panel is harder to classify (rough surface); the one on the bottom is easier to classify (smoother surface).

Definition 2: Let $\mu_j(x), 1 \leq j \leq C$ denote the membership value of instance x for class j and let $\mu(x) = [\mu_1(x), \ldots, \mu_C(x)]$. The local cooccurrence matrix of instance x is defined by

$$P(x) = \sum_{y, D_{xy} \leq r} \mu(x)^{\top} \mu(y). \tag{8}$$

The size of the local cooccurrence matrix is $C \times C$ (recall that C is the number of classes). It captures the distribution of instances around a specific instance. In other words, the element $p_{ij} = \sum_{y, D_{xy} \leq r} \mu_i(x) \mu_j(y)$ of matrix P shows the number of class j instances that are within the neighborhood r of instance x when instance x belongs to class i with membership $\mu_i(x)$.

Note, m_k child nodes are created when an attribute is used to split the instances at a given node. We can get the cooccurrence matrix for each branch (after the kth attribute is selected) by simply adding the cooccurrence matrix of all the instances along that branch and the overall cooccurrence matrix of the instances by summing the cooccurrence matrix along each branch.

Definition 3: Local coccurrence matrix after attribute k is selected

$$W^{(k)} = \sum_{i=1}^{m_k} \sum_{x} P(x) \tag{9}$$

[2]It is conceivable that there are other methods of evaluating the classifiability. For example, one can use the purity of the k-nearest neighbors of an instance. It is not immediately clear if any advantage is obtained by such a method over the one proposed here. Our choice of a texture based method however does conform to the intuitive notion of class label surface roughness as used here.

where x denotes any instance in the ith child node of current node.

Note that in the ideal case (perfect classifiability) matrix $W^{(k)}$ becomes a diagonal matrix. In general, the off-diagonal terms correspond to instances of different classes that occur with a neighborhood of radius r. We first normalize $W^{(k)}$ such that the sum of the elements of $W^{(k)}$ is one. Thus, we can define the classifiability, $L^{(k)}$ as

$$L^{(k)} = \sum_{i=1}^{C} w_{ii}^{(k)} - \sum_{i=1}^{C} \sum_{\substack{j=1 \\ j \neq i}}^{C} w_{ij}^{(k)} \tag{10}$$

where w_{ij} is an element in row i and column j of the matrix W.

The look-ahead term $L^{(k)}$ allows us to formulate an objective function that allows a decision to be made at a node so as to jointly optimize the node splitting criterion (information gain, GR, or any other chosen criterion) as well as classifiability of instances along all branches of the node. More specifically, we define the following objective function

$$J^{(k)} = B^{(k)} - \lambda L^{(k)} \tag{11}$$

where, $B^{(k)}$ can be equal to $E^{(k)}$ as defined in (4) or be equal to $\mathrm{GR}^{(k)}$ as defined in (5), and $L^{(k)}$ represents the look-ahead term discussed above. An attribute h selected to split the instances at a given node is then based on

$$h = \arg \min_{1 \leq k \leq n} J^{(k)}. \tag{12}$$

Equations (9)–(12) can then be used to choose an attribute to split the instances at a particular node. The resulting split in this case corresponds to one that maximizes the number of correct classifications at the node as well as facilitates the classification of the instances in each branch of the node. In the next section, we present some experimental results obtained using the proposed algorithm. Of course, when λ in (11) is 0, the proposed algorithm becomes identical to the F-ID3 algorithm or the GR algorithm (depending on what $B^{(k)}$ is defined as).

IV. Experimental Results

We present results with seven data sets to illustrate that the proposed look-ahead based fuzzy decision tree induction algorithm can produce smaller trees. Except for the first simulation, all data sets can be obtained from the University of California, Irvine, Machine Learning Repository [24] or from the restricted area of the UCI Machine Learning Repository [25].

For each of the data sets we obtain the performance of Fuzzy ID3 based on information gain by itself (IG), Fuzzy ID3 combined with the proposed method of look-ahead (IG+LA), GR by itself and GR combined with the proposed method of look-ahead (GR+LA). In general, it is possible to incorporate the proposed method of look-ahead with other node splitting criterion (for example, with the minimum classification ambiguity [4]).

We followed a consistent method of fuzzifying the data for all cases. When an attribute is categorical, the fuzzification is quite straightforward. We just treat each of the possible values of the attribute as a fuzzy subset. In this case, the membership value in a fuzzy subset is either zero or one. For numerical attributes, we use the k-means clustering algorithm to cluster the attribute values in to three clusters representing three linguistic terms T_1, T_2, T_3. The choice of the number of clusters is arbitrary though we were guided by the notion that a value can typically be thought of as being low, average, or high. We generate the memberships based on a triangular membership function and the three cluster centers (m_1, m_2, m_3 with $m_1 < m_2 < m_3$) obtained through k-means clustering. Specifically

$$\mu_{T_1} = \begin{cases} 1, & x \leq m_1 \\ (m_2 - x)/(m_2 - m_1), & m_1 < x < m_2 \\ 0, & x \geq m_2 \end{cases} \tag{13}$$

$$\mu_{T_2} = \begin{cases} 0, & x \geq m_3 \\ (m_3 - x)/(m_3 - m_2), & m_2 < x < m_3 \\ (x - m_1)/(m_2 - m_1), & m_1 < x < m_2 \\ 0, & x \leq m_1 \end{cases} \tag{14}$$

$$\mu_{T_3} = \begin{cases} 1, & x \geq m_3 \\ (x - m_2)/(m_3 - m_2), & m_2 < x < m_3 \\ 0, & x \leq m_2. \end{cases} \tag{15}$$

In all our simulations, the class label is *not* fuzzified, i.e., each instance belongs to a single class.

When separate training and testing data sets are present, results are reported based on construction of the decision tree with the training data and subsequent labeling of the instances in the testing data. When no separate training and testing data sets are present, we ran the algorithm three times. For each of the runs, we randomly divided the available data into two partitions—a training partition and a test partition. In this case, numbers reported are based on averaging over the three runs. The inference rule for the testing data is quite simple. Assume at some node, we choose attribute $A^{(i)}$ to split the data. When a test instance comes in, it goes to jth child node if it has the highest membership value in jth possible value of attribute $A^{(i)}$. This means that the fuzzy inference is actually the same as the nonfuzzy inference with the nonfuzzy partition for continuous features using three intervals: $[-\infty, (m_1 + m_2)/2], [(m_1 + m_2)/2, (m_2 + m_3)/2), [(m_2 + m_3)/2, +\infty]$. These intervals correspond to the three membership functions defined in (13)–(15). We repeat this procedure until we arrive at a leaf node where we classify the instance according to the class label at that leaf node.

A. Data Set I: The Illustrative Example of Section II

The first simulation is based on the example data set of Section II. To decide on the attribute to select based on the proposed method, we first obtain the local cooccurrence matrix and classifiability of the instances along the branches of the root node using (9) and (10) for each of the attributes as

$$W(\text{Outlook}) = W^{(1)} = \begin{bmatrix} 0.2336 & 0.0980 & 0.0865 \\ 0.0980 & 0.1138 & 0.0536 \\ 0.0865 & 0.0536 & 0.1762 \end{bmatrix}$$

and $L(\text{Outlook}) = L^{(1)} = 0.0472.$

$$W(\text{Temperature}) = W^{(2)} = \begin{bmatrix} 0.1543 & 0.0983 & 0.0906 \\ 0.0983 & 0.1548 & 0.0442 \\ 0.0906 & 0.0442 & 0.2249 \end{bmatrix}$$

and $L(\text{Temperature}) = 0.0678.$

TABLE II
EXPERIMENTAL RESULTS. TRAINING AND TESTING ACCUARCY IS EXPRESSED IN PERCENTAGES. NODES IN THE DECISION TREE WERE ADDED UNTIL FEWER THANERRORTOLERANCE INSTANCES ARE IN ERROR. WHEN THE CLASS LABLE IS CRISP, THE SUM OF THE INCONSISTANT CARDINALITIES EQUALS THE NUMBER OF INSTANCES MISCLASSIFIED. PERFORMANCE IS REPORTED FOR IG, IG+LA (INFORMATION GAIN WITH THE PROPOSED MEHTOD OF LOOK-AHEAD), GR, AND GR+LA (GAIN RATION WITH THE PROPOSED METHOD OF LOOK-AHEAD) BASED ALGORITHMS. .

Data Set	Parameters		IG	IG+LA	GR	GR+LA
Illustrative Example	$\lambda = 1$	# of Nodes	20	14	19	16
	$r = 3$	Training Acc.	N/A	N/A	N/A	N/A
	ET = 1	Testing Acc.	N/A	N/A	N/A	N/A
MONK-1	$\lambda = 1$	# of Nodes	85	41	84	41
	$r = 3$	Training Acc.	100.0	100.0	100.0	100.0
	ET = 0	Testing Acc.	86.75	100.0	84.72	100.0
MONK-2	$\lambda = 1$	# of Nodes	54	46	62	43
	$r = 3$	Training Acc.	80.47	82.25	81.06	81.66
	ET = 2	Testing Acc.	63.42	63.65	64.81	66.51
MONK-3	$\lambda = 1$	# of Nodes	45	42	42	39
	$r = 3$	Training Acc.	100.0	100.0	100.0	100.0
	ET = 0	Testing Acc.	94.4	94.9	90.28	90.74
Breast Cancer	$\lambda = 0.5$	# of Nodes	69	54	67	51
	$r = 3$	Training Acc.	98.56	98.56	98.56	98.56
	ET = 2	Testing Acc.	65.22	66.67	65.21	66.40
Wisconsin Breast Cancer	$\lambda = 1$	# of Nodes	135	109	139	103
	$r = 4$	Training Acc.	98.17	98.51	98.24	98.53
	ET = 2	Testing Acc.	94.36	94.19	94.74	94.85
Glass Identification	$\lambda = 1$	# of Nodes	176	148	172	146
	$r = 4$	Training Acc.	95.5	96.1	95.6	96.1
	ET = 2	Testing Acc.	92.86	93.09	92.86	93.10

$$W(\text{Humidity}) = W^{(3)} = \begin{bmatrix} 0.2108 & 0.0931 & 0.0871 \\ 0.0931 & 0.0985 & 0.0443 \\ 0.0871 & 0.0443 & 0.2419 \end{bmatrix}$$

and $L(\text{Humidity}) = L^{(3)} = 0.1022.$

$$W(\text{Wind}) = W^{(4)} = \begin{bmatrix} 0.2302 & 0.0912 & 0.0874 \\ 0.0912 & 0.0746 & 0.0370 \\ 0.0874 & 0.0370 & 0.2623 \end{bmatrix}$$

and $L(\text{Wind}) = 0.1377.$

From the above, it is clear that the use of the attribute `Wind` results in much greater classifiability. Using (11) and (12) we also obtain the attribute `Wind` as the attribute of choice for the root node. Proceeding in this way, we obtain the decision tree shown in bottom panel of Fig. 1. Comparing it to the original decision tree, it is clear that the one obtained using the proposed algorithm is significantly smaller.

B. MONK's Problem

For the next three simulations, we consider the well known MONK's data sets available at the UCI Machine Learning Repository [24]. The MONK's data sets are actually three subproblems. The domains for all MONK's problems are the same. There are 432 instances that belong to two classes and each instance is described by seven attributes $(a1, \ldots, a7)$. Among the seven attributes, there is one ID attribute (a unique symbol for each instance), which is not related to classification and is ignored in our simulations.

Data Set II: MONK-1 Problem: The target concept associated with the MONK-1 problem is $(a1 == a2)$ or $(a5 == 1)$. Table II summarizes the results obtained. It is clear that the use of look-ahead results in better performance. F-ID3 has 85 nodes while F-ID3+LA has only 41 nodes. The results with GR and GR+LA are similar. The testing accuracy is also better.

Data Set III: MONK-2 Problem: The target concept associated with the MONK-2 problem is: exactly two of $(a1 == 1, a2 == 1, a3 == 1, a4 == 1, a5 == 1, a6 == 1)$. Table II shows the results obtained. As in the previous case, the use of the proposed look-ahead method results in a considerable improvement.

Data Set IV: MONK-3 Problem: The target concept associated with the MONK-3 problem is ($a5 == 3$ and $a4 == 1$) or ($a5 \neq 1$ and $a2 \neq 3$). 5% noise is added to the training set. Results obtained are shown in Table II.

In this case, there is only a slight improvement. This is because of the agreement of the first term and the second

TABLE III
EFFECT OF VARYING λ ON THE IG+LA AND GR+LA ALGORITHMS FOR THE MONK-1 DATA SET.

		IG+LA			GR+LA		
r	λ	# Nodes	Tr. Acc.	Test. Acc.	# Nodes	Tr. Acc.	Test. Acc.
3.0	0.5	41	100.0	100.0	41	100.0	100.0
	1.0	41	100.0	100.0	41	100.0	100.0
	2.0	41	100.0	100.0	41	100.0	100.0
	3.0	41	100.0	100.0	41	100.0	100.0

TABLE IV
EFFECT OF VARYING r ON THE IG+LA AND GR+LA ALGORITHMS FOR THE MONK-1 DATA SET.

		IG+LA			GR+LA		
λ	r	# Nodes	Tr. Acc.	Test. Acc.	# Nodes	Tr. Acc.	Test. Acc.
1.0	1.0	85	100.0	86.57	84	100.0	84.72
	2.0	41	100.0	100.0	41	100.0	100.0
	3.0	41	100.0	100.0	41	100.0	100.0
	4.0	41	100.0	100.0	41	100.0	100.0

term of (11). In other words, an attribute that has highest information gain, happens to have the highest classifiability. This, however, is usually not the case.

C. Data Set V: Breast Cancer Data

For the next simulation we consider a real world dataset, i.e., the breast cancer data available in the restricted area of the UCI Machine Learning Repository [25]. The dataset has 286 instances that are described by nine attributes and belong to two classes: no-recurrence-events and recurrence-events. There are a total of nine instances that have missing attribute values. We removed those nine instances and used the remaining 277 instances. As there is no separate testing data in this case, we used 70% of the data for training and 30% of the data for testing. The algorithm was run thrice, each time with a randomly sampled training data set (70% of the total data) and a testing data set (30% of the total data). Average of the results obtained are shown in Table II and reflect both a smaller decision tree obtained as well as a slight improvement in test performance.

D. Data Set VI: Wisconsin Breast Cancer Data

For the following simulation, we took the Wisconsin Breast Cancer data. Each pattern in the data set has nine inputs and an associated class label (benign or malignant). The two classes are known to be linearly inseparable. The total number of instances are 699 (458 benign, and 241 malignant), of which 16 instances have a single missing attribute. We removed those 16 instances and used the remaining 683 instances. As there is no separate testing data in this case, we used 50% of the data for training and 50% of the data for testing. In part, the reason for the equal split in this case as opposed to the 70%–30% split of the previous simulation is that there are a larger number of instances and using 50% of the data for training resulted in a sufficiently large training data set. As before, the algorithm was run thrice, each time with a randomly sampled training data set (50% of the total data) and a testing data set (50% of the total data). Average of the results obtained are shown in Table II.

E. Data Set VII: Glass Identification Data

For the final simulation, we consider the glass identification data set. There are totally 214 instances and each instance is described by ten attributes (including an Id attribute). All attributes are continuous valued. There are two classes: window glass and nonwindow glass. As there is no separate testing data in this case, we used 70% of the data for training and 30% of the data for testing. As before, the algorithm was run thrice, each time with a randomly sampled training data set (70% of the total data) and a testing data set (30% of the total data). The results obtained are shown in Table II.

A final comment about the simulations. In general, there is no definite way of knowing an appropriate value to use for r [see (8)] and λ [see (11)]. Increasing the value of λ results in a larger emphasis being placed on the classifiability term and a smaller emphasis being placed on the information gain (or GR) term. In general, we found $\lambda = 1$ to consistently provide good results. In all our simulations λ was fixed at one. r in general should increase linearly with n, the number of attributes. Indeed, r should be large enough such that at least a few instances are present within that neighborhood of each instance. r should also be small enough such that classifiability is evaluated locally. Table III shows the effect of varying λ for the MONK-1 data set when IG+LA is used and when GR+LA is used and Table IV shows the effect of varying r for the same data set. As these tables show there exist a considerable range of values for r and λ for which the same results are obtained. However, as the first row of Table IV shows that when r is small then it is possible that there are not enough instances in the neighborhood of an instance making it difficult to evaluate classifiability. In that case, the algorithm degrades to one without look-ahead, i.e., plain IG or the GR algorithms.

V. Conclusion

In this paper we presented a novel approach for evaluating the classifiability of instances based on evaluating the texture of the class label surface. Based on that, we also proposed an algorithm for fuzzy decision tree induction using look-ahead. The proposed method of look-ahead when combined with either information gain or with gain ratio outperforms standalone information gain and gain ratio respectively. More specifically, it results in smaller decision trees and as a consequence better generalization (test) performance.

The proposed algorithm does require additional computation when compared to F-ID3 or GR. In particular, the look-ahead term $L^{(k)}$ requires finding instances that are within a distance r from a given instance. The inter-instance distance, however, can be computed once, stored, and does not need to be computed at each node. This additional time may well be justified in most situations in light of the considerably improved performance that results from the decision trees constructed using the proposed algorithm.

The method of look-ahead proposed in this paper can also be used in crisp decision tree induction. Within the context of crisp decision tree induction, it is common to use a linear discriminant at each node. One step, or a few-step look-ahead, when used, is also based on a linear discriminant. Since exhaustive look-ahead is not feasible, it is conceivable that entirely different results are obtained with a $(d+1)$-step look-ahead when compared to those obtained using a d-step look-ahead. In addition, the use of a linear discriminant in doing the look-ahead implicitly assumes a model that is satisfied by the data distribution. The proposed method of look-ahead is nonparametric and therefore does not suffer from the bias of an assumed model.

References

[1] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, pp. 81–106, 1986.

[2] ——, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1993.

[3] R. L. P. Chang and T. Pavlidis, "Fuzzy decision tree algorithms," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-7, pp. 28–35, 1977.

[4] Y. Yuan and M. J. Shaw, "Induction of fuzzy decision trees," *Fuzzy Sets Syst.*, vol. 69, pp. 125–139, 1995.

[5] K. J. Cios and L. M. Sztandera, "Continuous ID3 algorithm with fuzzy entropy measures," *Proc. IEEE Int. Conf. Fuzzy Syst.*, pp. 469–476, 1992.

[6] C. Z. Janikow, "Exemplar learning in fuzzy decision trees," *Proc. FUZZ-IEEE*, pp. 1500–1505, 1996.

[7] ——, "Fuzzy decision trees: Issues and methods," *IEEE Trans. Syst., Man, Cybern.*, vol. 28, no. 1, pp. 1–14, 1998.

[8] L. X. Wang and J. M. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Trans. Syst. Man, Cybern.*, vol. 22, pp. 1414–1427, 1992.

[9] L. M. Sztandera, "Fuzzy neural trees," *Encyclopedia Comput. Sci. Technol.*, vol. 40, no. 25, pp. 87–104, 1999.

[10] K. J. Cios and L. M. Sztandera, "Ontogenic neuro-fuzzy algorithm: F-CID3," *Neurocomputing*, vol. 14, no. 4, pp. 383–402, 1997.

[11] L. M. Sztandera, "Fuzzy neural trees," *Inf. Sci.* , vol. 90, no. 1/4, pp. 155–177, 1996.

[12] L. Breiman, J. H. Friedman, J. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Belmont, CA: Wadsworth , 1984.

[13] A. Hart, *Research and Developments in Expert Systems*, M. Bramer, Ed, Cambridge: Cambridge University Press, 1984. .

[14] J. Mingers, "Expert systems—Experiments with rule induction," *J. Oper. Res.*, vol. 38, pp. 39–47, 1987.

[15] ——, "An empirical comparison of selection measures for decision-tree induction," *Machine Learning*, vol. 3, pp. 319–342, 1989.

[16] W. Buntine and T. Niblett, "A further comparison of splitting rules for decision-tree induction," *Machine Learning*, vol. 8, pp. 75–85, 1989.

[17] J. R. Quinlan, "Simplifying decision trees," *Int. J. Man-Machine Studies*, vol. 27, pp. 221–234, 1987.

[18] X. Wang, B. Chen, G. Qian, and F. Ye, "On the optimization of fuzzy decision trees," *Fuzzy Sets Syst.*, vol. 112, pp. 117–125, 2000.

[19] J. R. Quinlan and R. M. Cameron-Jones, "Oversearching and layered search in empirical learning," in *Proc. 14th Int. Conf. Artificial Intelligence*. San Mateo, California, 1995, pp. 1019–1024.

[20] J. F. Elder, "Heuristic search for model structure," in *Learning from Data: Artificial Intelligence and Statistics V, Lecture Notes in Statistics*, D. Fischer and H.-J. Lenz, Eds. Vienna, Austria, Germany: Springer-Verlag, 1995, vol. 112, pp. 131–142.

[21] S. K. Murthy and S. Salzberg, "Lookahead and pathology in decision tree induction," in *Proc. 14th Int. Conf. Artificial Intell.*. San Mateo, CA, 1995, pp. 1025–1031.

[22] R. Kothari and M. Dong, *Lecture Notes in Pattern Recognition*, S. K. Pal and A. Pal, Eds, Singapore: World Scientific , 2000.

[23] R. M. Haralick and L. G. Shapiro, *Computer Robot Vision*. Reading, MA: Addison-Wesley, 1992.

[24] C. J. Merz and P. M. Murphy. (1996) UCI Repository of Machine Learning Databases. Department of Information and Computer Science, University of California, Irvine, CA. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

[25] ——, (1996) UCI Repository of Machine Learning Databases. Department of Information and Computer Science, University of California, Irvine, CA. [Online]. Available: http:// www.ics.uci.edu/~mlearn/ML-Repository.html