Pattern Trees Induction: A New Machine Learning Method

Zhiheng Huang, Tamás D. Gedeon, and Masoud Nikravesh

***Abstract*—Fuzzy classification is one of the most important applications in fuzzy set and fuzzy-logic-related research. Its goal is to find a set of fuzzy rules that form a classification model. Most of the existing fuzzy rule induction methods (e.g., the fuzzy decision trees (FDTs) induction method) focus on searching rules consisting of triangular norms (t-norms) (i.e., AND) only, but not triangular conorms (t-conorms) (OR) explicitly. This may lead to the omission of generating important rules that involve t-conorms explicitly. This paper proposes a type of tree termed *pattern trees* (PTs) that makes use of different aggregations, including both t-norms and t-conorms. Like decision trees, PTs are an effective tool for classification applications. This paper discusses the difference between decision trees and PTs, and also shows that the subsethood-based method (SBM) and the weighted-subsethood-based method (WSBM) are two specific cases of PT induction. A novel PT induction method is proposed using similarity measure and fuzzy aggregations. The comparison to other classification methods including SBM, WSBM, C4.5, nearest neighbor, support vector machine, and FDT induction shows that: 1) PTs can obtain high accuracy rates in classifications; 2) PTs are robust to overfitting; and 3) PTs, especially simple pattern trees (SPTs), maintain compact tree structures.**

***Index Terms*—Fuzzy classification model, fuzzy decision trees (FDTs), pattern trees (PTs).**

I. Introduction

THE MAJOR advantage of using fuzzy rules for classification applications is to maintain transparency as well as a high accuracy rate. There is extensive research on fuzzy-set-based machine learning. Wang and Mendel [15] have presented an algorithm for generating fuzzy rules by learning from examples. Inspired by the classic decision tree induction by Quinlan [10], there is substantial work on fuzzy decision trees (FDTs). For example, Yuan and Shaw [20] have proposed FDTs induction using fuzzy entropy. Janikow [5], Olaru and Wehenkel [8] have presented different FDT inductions. Suárez and Lutsko [14] and Wang *et al.* [16] have presented optimizations of FDTs. Other fuzzy-based machine learning algorithms are also suggested. For example, Chen *et al.* [3] have proposed a subsethood-based machine learning method. Rasmani and Shen [12] have proposed a weighted-fuzzy-subsethood-based learning method. To avoid the exponential growth of the size of the rule base when the number of input variables increases, Raju *et al.* [11] have proposed hierarchical fuzzy systems. Recently, Kóczy *et al.* [6] and Wong *et al.* [18] have presented fuzzy signatures that model the complex structure of the data points in a hierarchical manner.

Most of the existing fuzzy rule induction methods, including FDTs [20], focus on searching for rules that only use triangular norms (t-norms) operators [13] such as the MIN and algebraic MIN. Disregarding of the triangular conorms (t-conorms) such as MAX and algebraic MAX is due to the fact that any rule using t-conorms can be represented by several rules that use t-norms only. This is certainly true, and it is helpful to simplify the rule induction process by only considering t-norms. However, it may fail to generate important rules in which fuzzy terms are explicitly connected with t-conorms. This will be shown in an artificial dataset in Section IV. Related research includes fuzzy signature and multiaggregator FDTs that employ varied operators. Kóczy *et al.* [6] have proposed fuzzy signatures to model the complex structures of the data points using different aggregation operators including MIN, MAX, and average, etc. Nikravesh [7] has presented evolutionary computation (EC) based multiple aggregator FDTs, in which more choices of operators, such as harmonic mean, are available.

This paper proposes a type of tree termed *pattern trees* (PTs) that makes use of different aggregations (including t-norms and t-conorms). Like decision trees, PTs serve as a tool for classification applications. A novel PT induction method is proposed using similarity measure and fuzzy aggregations. The comparison to other classification methods, including subsethood-based method (SBM), weighted-subsethood-based method (WSBM), C4.5, nearest neighbor, support vector machine (SVM), and FDT induction, shows that: 1) PTs can obtain high accuracy rates in classifications; 2) PTs are robust to overfitting; and 3) PTs, especially simple PTs, maintain compact tree structures.

The rest of the paper is arranged as follows. Section II provides the definitions of similarity, aggregations and PTs. In Section III, a novel PT induction method using similarity measures and fuzzy aggregations is proposed. Section IV presents the related work that includes the SBM, WSBM, and FDTs. Section V gives the experimental results. Finally, Section VI concludes the paper and points out some further research work.

II. Similarity, Aggregations, and Pattern Trees

This section first defines the similarity and fuzzy aggregations that will be used in PTs induction in Section III. It then presents the concept of PTs, the aggregations of PTs, and the use of PTs as a classification tool.

Manuscript received August 29, 2006; revised October 30, 2007; accepted December 22, 2007.

Z. Huang and M. Nikravesh are with the Department of Electrical Engineering and Computer Science, University of California at Berkeley, Berkeley, CA 94720-1776 USA (e-mail: zhiheng@cs.berkeley.edu; nikravesh@cs.berkeley.edu).

T. D. Gedeon is with the Department of Computer Science, Australian National University, Canberra, A.C.T. 0200, Australia (e-mail: tom@cs.anu.edu.au).

Color version of one or more of the figures in this paper are available at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TFUZZ.2008.924348

TABLE I
SIMILARITY MEASURES

Name	Definition
Simple matching	$A \cap B$
Jaccard	$\frac{A \cap B}{A \cup B}$
Dice	$2\frac{A \cap B}{A + B}$

TABLE II
ARTIFICIAL DATASET

A		B		Class	
A_1	A_2	B_1	B_2	X	Y
0.8	0.2	0.0	1.0	0.9	0.1
0.9	0.1	0.1	0.9	0.8	0.2
0.5	0.5	0.9	0.1	0.7	0.3
0.2	0.8	0.8	0.2	0.9	0.1
0.4	0.6	0.4	0.6	0.3	0.7
0.3	0.7	0.5	0.5	0.1	0.9

A. *Similarity Between Two Fuzzy Terms*

Let A and B be two fuzzy sets [21] defined on the universe of discourse U. The commonly used fuzzy similarity definitions are shown in Table I, where $\cap$ and $\cup$ denote a certain t-norm operator and a t-conorm, respectively. Usually, the MIN ($\wedge$) and MAX ($\vee$) operators are used. Jaccard measure, the most commonly used, is selected to demonstrate the construction of PTs in Section III, which is computed in practice as

$$\mathrm{Sim}(A,B) = \frac{\sum_{j=1}^{m}[\mu_A(x_j) \wedge \mu_B(x_j)]}{\sum_{j=1}^{m}[\mu_A(x_j) \vee \mu_B(x_j)]} \qquad (1)$$

where $x_j, j = 1, \ldots, m$, are the crisp values discretized in the variable domain, and $\mu_A(x_j)$ and $\mu_B(x_j)$ are the fuzzy membership values of x_j for A and B.

An alternative similarity definition is proposed in this paper for PT construction. Consider that the root mean square error (RMSE) of fuzzy sets A and B can be computed as

$$\mathrm{RMSE}(A,B) = \sqrt{\frac{\sum_{j=1}^{m}(\mu_A(x_j) - \mu_B(x_j))^2}{m}}. \qquad (2)$$

The RMSE-based fuzzy set similarity can thus be defined as

$$\mathrm{Sim}(A,B) = 1 - \mathrm{RMSE}(A,B). \qquad (3)$$

The larger the value $\mathrm{Sim}(A,B)$ takes, the more similar A and B are. It is easy to verify that both Jaccard and RMSE-based similarity definitions retain $0 \leq \mathrm{Sim}(A,B) \leq 1$.

Example 1: Assume that an artificial dataset, as shown in Table II, has two input variables A and B, with each having two fuzzy linguistic terms A_i and B_i, $i = 1, 2$. Also assume that this dataset has two output classes X and Y. The similarity between A_1 and X, according to Jaccard, is computed as $\mathrm{Sim}(A_1, X) = 0.6585$, and, according to RMSE, is computed as $\mathrm{Sim}(A_1, X) = 0.6838$.

B. *Fuzzy Aggregations*

Fuzzy aggregations are logic operators applied to fuzzy membership values or fuzzy sets. They have three subcategories, namely t-norm, t-conorm, and averaging operators such as weighted averaging (WA) and ordered-WA (OWA) [19]. t-norms were introduced by Schweizer and Sklar [13] to model distances in probabilistic metric spaces. In fuzzy sets theory, t-norms and t-conorms are extensively used to model logical operators AND and OR. The basic t-norm and t-conorm pairs that operate on two fuzzy membership values a and b, $a, b \in [0, 1]$, are shown in Table III. The definitions of WA and OWA are given as follows.

TABLE III
BASIC t-NORMS AND t-CONORMS PAIRS

Name	t-norm	t-conorm
MIN/MAX	$min\{a,b\} = a \wedge b$	$max\{a,b\} = a \vee b$
Algebraic AND/OR	ab	$a + b - ab$
Łukasiewicz	$max\{a+b-1, 0\}$	$min\{a+b, 1\}$
EINSTEIN	$\frac{ab}{2-(a+b-ab)}$	$\frac{a+b}{1+ab}$

Definition 1: A WA operator of dimension n is a mapping $R^n \rightarrow R$ that has an associated n-element vector $w = (w_1, w_2, \ldots, w_n)^T, w_i \in [0,1], 1 \leq i \leq n$, and $\sum_{i=1}^{n} w_i = 1$, so that

$$\mathrm{WA}(a_1, \ldots, a_n) = \sum_{j=1}^{n} w_j a_j. \qquad (4)$$

Definition 2: An OWA operator [19] of dimension n is a mapping $R^n \rightarrow R$ that has an associated n-element vector $w = (w_1, w_2, \ldots, w_n)^T, w_i \in [0,1], 1 \leq i \leq n$, and $\sum_{i=1}^{n} w_i = 1$, so that

$$\mathrm{OWA}(a_1, \ldots, a_n) = \sum_{j=1}^{n}(w_j f_j(a_1, \ldots, a_n)) \qquad (5)$$

where $f_j(a_1, \ldots, a_n)$ returns the jth largest element of the collection $\{a_1, \ldots, a_n\}$.

A fundamental difference of OWA from WA aggregation is that the former does not have a particular weight w_i associated for an element, but rather a weight is associated with a particular ordered position of the element. It is worth noting that two special OWA operators are equal to MAX and MIN:

1) If $w^* = (1, 0, \ldots, 0)$, then

$$\mathrm{OWA}(a_1, \ldots, a_n) = \max\{a_1, \ldots, a_n\}. \qquad (6)$$

2) If $w_* = (0, 0, \ldots, 1)$, then

$$\mathrm{OWA}(a_1, \ldots, a_n) = \min\{a_1, \ldots, a_n\}. \qquad (7)$$

The main factor in determining which aggregation should be used is the relationship between the criteria involved. Compensation has the property that a higher degree of satisfaction of one of the criteria can compensate for a lower degree of satisfaction of another criterion. The variable w^* means full compensation (OR) and w_* means no compensation (AND). Normally, an OWA operator lies in between these two extremes. An OWA operator with many nonzero weights near the top will be more OR than AND.

The extension of aforementioned aggregations to fuzzy terms is straightforward: the aggregation result of two fuzzy terms would be a new fuzzy term, with the aggregation applied between two fuzzy terms in a pairwise fashion.

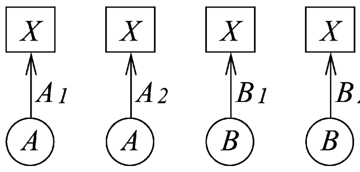


Fig. 1. Primitive PTs.

Example 2: Consider the small dataset in Example 1. The MIN of fuzzy terms A_1 and A_2 is calculated as

$$\text{MIN}(A_1, A_2) = (0.2, 0.1, 0.5, 0.2, 0.4, 0.3)^T$$

while the OWA of fuzzy terms A_1 and A_2 is calculated as the following, if the weight vector of $w = (0.7, 0.3)$ is given by

$$\text{OWA}(A_1, A_2) = (0.6, 0.7, 0.5, 0.6, 0.5, 0.6)^T.$$

C. Pattern Trees

A PT is used to represent the pattern of data that belongs to the *same* class. In PT structures, leaf nodes represent fuzzy terms and internal nodes represent fuzzy aggregations. Under the same conditions in Example 1, Figs. 1 and 3 show some exemplar PTs. As can be seen, a PT is a tree that propagates fuzzy terms using different fuzzy aggregations. Each PT represents a structure for one output class that is located at the top as the root of this tree. The fuzzy terms of input variables are on different depths (except the top) and they use fuzzy aggregations as presented in Section II-B to aggregate upwards. In particular, each of the trees shown in Fig. 1 consists of only one leaf node (fuzzy set) to predict the output class (X in this case) and they are referred to as *primitive PTs*. Typically, primitive PTs do not lead to high classification accuracy. Multidepth PTs that are built via the aggregation of primitive trees are desirable to achieve satisfactory performance.

Aggregation of two PTs consists of three parts: the *candidate PT*, the *slave PT*, and the aggregation operator. The slave tree is used to grow the candidate tree by attaching the root of the slave tree to the root of the candidate tree using the given aggregation: the aggregated tree thus subsumes both the candidate tree and the slave tree. The names of candidate and slave trees come from the assumption that the latter cannot have more depth than the former. This paper follows the convention that a candidate tree is always drawn as the left child of the aggregated tree while a slave tree is drawn as the right child. Fig. 2 shows two examples of PT aggregation, with the first being the aggregation between the candidate tree $B_1 \Rightarrow X$ and the slave tree $A_2 \Rightarrow X$ using the AND operator, and the second being the aggregation between the previously aggregated tree and the slave tree $A_1 \Rightarrow X$ using the OR operator.

From fuzzy term aggregation described in Section II-B, the aggregated fuzzy term for an aggregated PT can be computed, which consists of a vector of fuzzy membership values and is denoted as *tree term* T (for the aggregated PT) for late reference. Note that we also use T to denote a PT; the notation of the fuzzy tree term and the tree itself is thus interchangeable.

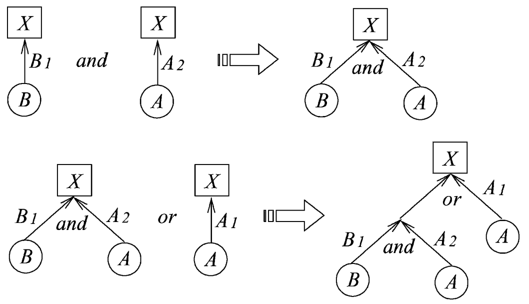


Fig. 2. Aggregations of PTs.

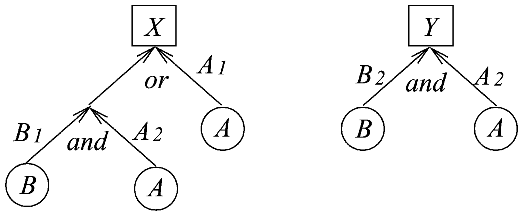


Fig. 3. Two example PTs.

Example 3: Let us consider the small dataset in Example 1. The first aggregation in Fig. 2 leads to the tree

$$\begin{aligned} T &= \text{MIN}(B_1, A_2) \\ &= (0.0, 0.1, 0.5, 0.8, 0.4, 0.5)^T \end{aligned}$$

and the second leads to the tree

$$\begin{aligned} T &= \text{MAX}(\text{ MIN }(B_1, A_2), A_1) \\ &= (0.8, 0.9, 0.5, 0.8, 0.4, 0.5). \end{aligned}$$

Aggregations of small PTs to complex ones may help increase the goodness of trees, thus resulting in high classification accuracy. The goodness of a PT is defined as the similarity between the tree term T and the output class that this tree represents. This definition is motivated by the fact that the more similar the tree and the output class, the more accurate prediction the tree can offer for data points with respect to the class that the tree represents.

Example 4: Consider the small dataset in Table II and the two aggregated PTs in Fig. 2. The goodness (similarities) of the first and second trees according to (3) are 0.50 and 0.80, respectively. That means the second PT does a better job of predicting the class X than the first one.

For a classification application that involves several output classes, the worked model should have as many PTs as the number of output classes, with each PT representing one class. For our problem in classifying to X or Y class, two PTs representing X and Y, respectively, are required. Fig. 3 shows an example model that consists of two such trees. When a new data sample is tested over a PT, it traverses from the bottom to the top and finishes with a truth value, indicating the degree to which this data sample belongs to the output class of this PT. The output class with the maximal truth value is chosen as the prediction class.

Algorithm 1 Induction of a simple pattern tree

```
1: Comment: Initialization
2: $\boldsymbol{P} = \{A_{ij}\}, i = 1, \ldots, n; j = 1, \ldots, m$
3: $C_0 = \mathrm{argmax}_{P \in \boldsymbol{P}}[Sim(P, X_0)]$
4: Comment: Induction
5: for $k = 1$ to $d$ step 1 do
6:     $\{\psi_k, S_k\} = \mathrm{argmax}_{\psi \in \Psi, S \in \boldsymbol{P}, S \notin C_{k-1}}[Sim(C_{k-1}\psi S, X_0)]$
7:     $C_k = C_{k-1}\psi_k S_k$
8:     if $Sim(C_k, X_0) < Sim(C_{k-1}, X_0)$ then
9:         k = k-1
10:        break
11:    end if
12: end for
13: return $C_k$
```

Example 5: Consider classifying the first fuzzy data in Table II, $A_1 = 0.8, A_2 = 0.2, B_1 = 0.0$, and $B_2 = 1.0$, using the PTs as shown in Fig. 3; as the truth values of these data over PTs for classes X and Y are 0.8 and 0.2, respectively, X is chosen as the output class.

Conventional fuzzy rules can be extracted from PTs. For example, the following rules can be obtained from the PTs in Fig. 3:

$$\text{Rule 1 : IF } A = A_1 \text{ THEN class } = X \tag{8}$$

$$\text{Rule 2 : IF } A = A_2 \text{ AND } B = B_1 \text{ THEN class } = X \tag{9}$$

$$\text{Rule 3 : IF } A = A_2 \text{ AND } B = B_2 \text{ THEN class } = Y. \tag{10}$$

It is worth noting that, in addition to the conventionally used fuzzy aggregations MIN/MAX, PTs can use any aggregations, as described in Section II-B.

III. Proposed Pattern Tree Induction Method

Without losing generality, assume that a dataset has n input variables $A_i, i = 1, 2, \ldots, n$, and one output variable X. Further assume that each of the input variables has m fuzzy linguistic terms denoted by $A_{ij}, i = 1, 2, \ldots, n$, and $j = 1, 2, \ldots, m$, and the output variable has k fuzzy or linguistic terms denoted by $X_j, j = 1, 2, \ldots, k$. That is, each data point is represented by a fuzzy membership value vector of dimension $(nm + k)$. The task is to build k PTs for the k output classes (fuzzy or linguistic terms).

The induction of a PT, say for class X_0, is described in Algorithm 1. The induction of other PTs follows the same principle. In the initialization, the set of primitive trees $\boldsymbol{P}$ is constructed, in which each fuzzy term $A_{ij}, i = 1, \ldots, n, j = 1, \ldots, m$, is used to construct a primitive PT. The primitive PT that has the highest similarity to output class term X_0 is then selected as the initial candidate tree C_0. Note that the bold symbol such as $\boldsymbol{P}$ indicates that it contains a set of trees in contrast to one tree such as C_0. The subscript of zero in C_0 indicates that the tree has zero depth.[1] In induction, the aggregation is attempted between the previous candidate tree C_{k-1} and any

[1]Here we assume that a primitive PT has zero depth. The depth of a nonprimitive tree is the maximum length of branches in such a tree.

Algorithm 2 Induction of a general pattern tree

```
1: Comment: Initialization
2: $\boldsymbol{P} = \{A_{ij}\}, i = 1, \ldots, n; j = 1, \ldots, m$
3: $\boldsymbol{C}_0 = \mathrm{argmaxL}_{P \in \boldsymbol{P}}[Sim(P, X_0)]$
4: $\boldsymbol{S}_0 = null$
5: Comment: Induction
6: for $k = 1$ to $d$ step 1 do
7:     $\{\{C_{k0}, \psi_{k0}, S_{k0}\}, \ldots, \{C_{k,L-1}, \psi_{k,L-1}, S_{k,L-1}\}\} =$
       $\mathrm{argmaxL}_{C \in \boldsymbol{C}_{k-1}, \psi \in \Psi, S \in \boldsymbol{P} \cup \boldsymbol{S}_{k-1}, S \notin C}[Sim(C\psi S, X_0)]$
8:     $\boldsymbol{C}_k = \{\{C_{k0}\psi_{k0}S_{k0}\}, \ldots, \{C_{k,L-1}\psi_{k,L-1}S_{k,L-1}\}\}$
9:     $\boldsymbol{S}_k = \mathrm{maxM}(\boldsymbol{S}_{k-1}, \boldsymbol{C}_k)$
10:    if $\max(\boldsymbol{C}_k) < \max(\boldsymbol{C}_{k-1})$ then
11:        k=k-1
12:        break
13:    end if
14: end for
15: return $\max(\boldsymbol{C}_k)$
```

primitive tree S in the primitive tree set $\boldsymbol{P}$, using any aggregation ψ drawn from the aggregation set Ψ. Note that when $\psi = \mathrm{WA}$ or $\psi = \mathrm{OWA}$, the weights that make the aggregated term most similar to class term are used. A constraint is imposed upon the aggregation: the primitive tree S cannot be a subset tree of the candidate tree C_{k-1}, which prevents a primitive tree from being used in the aggregated tree more than once. Among all the aggregated trees, the one that has the highest similarity to class term X_0 is selected as the current candidate tree C_k, which has one more depth than the previous candidate tree C_{k-1}. If the candidate tree has reached the predefined depth d,[2] or the new candidate tree C_k has a lower similarity to X_0 than the previous one, C_{k-1}, the induction stops and the tree that has the highest similarity is returned as the optimal tree.

Notice that in Algorithm 1, an aggregation always happens between a candidate tree (either a primitive tree or not) and a slave primitive tree. The aggregated trees thus always have one fuzzy term as its right child for any of the internal nodes. These kinds of trees are denoted as *simple pattern trees* (SPTs) [4]. In contrast, PTs that do not have such a constraint are referred to as *general PTs*. This paper proposes an induction method for general PTs, which subsumes the SPT induction in [4] as one specific case. In particular, there are two key differences between the induction of SPTs and that of general PTs: 1) the former keeps track of only one candidate tree in induction, while the latter can consider more than one and 2) the former uses only primitive trees to grow a candidate tree, while the latter does not have such a constraint. In order to perform the induction of general PTs, two sizes, namely the size of candidate trees L and the size of slave trees M, should be specified beforehand. They indicate the search space for candidate trees and slave trees, respectively. Intuitively, the higher the values of L and M, the more exhaustive the search of optimum PTs would be performed.

Algorithm 2 shows the induction of a general PT for class X_0 in pseudocode. In the initialization, the set of primitive trees $\boldsymbol{P}$

[2]The determination of d is a design choice; we make use of $d = 5$ or $d = 10$. In fact, as suggested by the experiments, $d = 5$ attains compact tree structures and $d = 10$ can achieve higher performance.

is constructed, in which each fuzzy term $A_{ij}, i = 1, \ldots, n, j = 1, \ldots, m$, is used to construct a primitive PT. The L primitive PTs that have the highest similarities to output class term X_0 are selected to form the set of initial candidate trees $\boldsymbol{C}_0$. Note that the bold symbol indicates $\boldsymbol{C}_0$, which consists of a set of trees rather than one tree. Again, the subscript of zero indicates that the trees in this set have zero depth. The slave tree set is empty at the initialization. In induction, the aggregation is attempted between any previous candidate tree C (in the candidate tree set $\boldsymbol{C}_{k-1}$) and any slave tree S in primitive tree set or slave tree set $\boldsymbol{P} \cup \boldsymbol{S}_{k-1}$, using any aggregation operator ψ drawn from the aggregation set Ψ. Again the constraint is imposed upon the aggregation: the slave tree cannot be a subset tree of the candidate tree. Since a candidate tree always has more than or at least equal to the depths of the slave tree, the aggregations lead to the candidate trees in $\boldsymbol{C}_k$ have one more depth than those in $\boldsymbol{C}_{k-1}$. Among all the aggregated trees, the best L ones are selected to form the new candidate tree set $\boldsymbol{C}_k$. The slave tree set $\boldsymbol{S}_k$ is then updated with the M trees that have the highest similarities to class term among the current candidate tree set $\boldsymbol{C}_k$ and previous slave tree set $\boldsymbol{S}_{k-1}$. If the candidate trees have reached the predefined depth d or the highest similarity of the current candidate trees is less than that of previous loop, the induction stops and the candidate tree that has the highest similarity is returned as the optimal tree.

In terms of computation complexity, both simple and general PT inductions are of $O(nmo)$, where n, m, and o are the number of input variables, the number of fuzzy terms for each variable, and the number of data. The computation effort is mostly spent on line 6 in Algorithm 1 and line 7 in Algorithm 2. In the SPT induction, there are nm slave trees used to aggregate a candidate tree in each *for* loop. In each aggregation, the complexity of $O(o)$ is required. Given that the predetermined depth d is a constant, the complexity is thus $O(nmo)$. Similarly for general PT induction, the complexity is aggregations between L candidate trees and nm primitive slave trees plus a constant number (M) of nonprimitive slave trees; thus, the complexity remains the same as $O(nmo)$.

Two examples of PTs induction are given as follows for clarification, with the first presenting the induction of a SPT and the second presenting the induction of a general PT.

Example 6: Assume that the artificial dataset in Table II is given. Fig. 4 shows the induction of a SPT for class X using only MIN/MAX, OWA, and WA aggregations (for simplicity). The construction of a tree for Y follows the same principle and is thus omitted. In this figure, each of the PTs are denoted as $T_i, i = 0, \ldots, 9$. They are numbered in the order of the time that the trees are constructed. That is, the trees with smaller indexes are constructed earlier than the ones with larger indexes. The candidate trees are highlighted by rectangular boxes with rounded corners. In particular, solid, dashed, and dotted lines are used for candidate trees with depths 0, 1, and 2, respectively (corresponding to the loops on line 5 in Algorithm 1). The number shown at the bottom of each tree T_i is the similarity between the tree and the class term X. The detailed computation is described as follows.

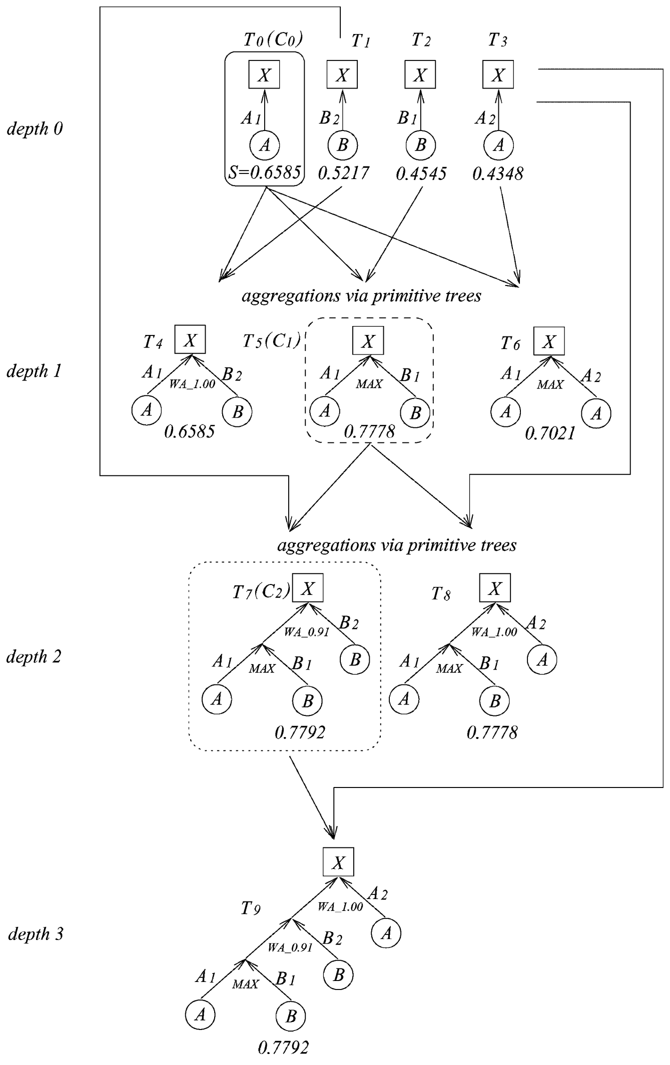


Fig. 4. Induction of a SPT.

TABLE IV
SIMILARITIES BETWEEN AGGREGATED TREES AND CLASS TERM X IN STEP 3

	T_1	T_2	T_3
MIN	0.5610	0.3000	0.3500
MAX	0.6087	0.7778	0.7021
OWA	0.6065	0.7740	0.6815
WA	0.6585	0.6543	0.6356

1) Build four primitive PTs T_0, T_1, T_2, and T_3, with the Jaccard similarities calculated from (1) being 0.6585, 0.5217, 0.4545, and 0.4348, respectively.
2) Assign T_0 that has the highest similarity to class term X as the candidate tree C_0 for depth zero trees.
3) Now consider the tree aggregation from depth 0 to depth 1. Try the rest of the primitive PTs T_1, T_2, and T_3 in turn to aggregate with candidate tree T_0 using different aggregation operators MIN, MAX, OWA, and WA. Note that T_0 is not allowed as a slave tree to aggregate the candidate tree $C_0 = T_0$ as it has already appeared in the candidate tree. The similarities between the aggregated trees and the class term X are shown in Table IV.

TABLE V
SIMILARITIES BETWEEN AGGREGATED TREES AND CLASS TERM X IN STEP 5

	T_1	T_3
MIN	0.5349	0.4762
MAX	0.7500	0.7143
OWA	0.7609	0.7109
WA	0.7792	0.7778

4) Choose the aggregated tree T_5 as the new candidate tree C_1 for depth 1 trees. This is because the aggregation of T_0 and T_2 with a MAX operator results in the highest similarity of 0.7778 that is greater than the similarity produced by T_0 alone. Also, the aggregated tree term of $T_5, (A_1 \text{ MAX } B_1) = \{0.8, 0.9, 0.9, 0.8, 0.4, 0.5\}$, is stored for further aggregation.
5) Now consider the tree aggregation from depth 1 to depth 2. Try the rest of the primitive trees T_1 and T_3 in turn to aggregate with T_5 using different aggregation operators. The similarities of aggregated trees are shown in Table V.
6) Choose the aggregated tree T_7 as new candidate tree C_2, since it has the highest similarity 0.7792 among all aggregated trees, and it is also greater than the similarity produced by T_5. Note that the weights $w = (0.91, 0.09)$ is calculated, associated with the WA aggregation. Also, the aggregated term of $T_7 = \{0.8185, 0.9000, 0.8259, 0.7444, 0.41856, 0.50000\}$ is stored for further aggregation.
7) Now consider the tree aggregation from depth 2 to depth 3. Try the rest of the primitive tree T_3 to aggregate with T_7 using different aggregation operators. None of the similarities of aggregated trees are greater than the current similarity of 0.7792. The PT induction stops and the best candidate tree C_2 is returned.

It is worth noting that each primitive tree is being checked to see if it is a subset tree of the candidate tree before the aggregation happens. For example, at depth 1 for candidate tree T_5, only T_1 and T_3 among all primitive trees are allowed to aggregate. This is because other primitive trees have already appeared in T_5.

Example 7: Following the conditions given in the previous example, the induction of a general PT using the size of candidate trees $L = 2$ and size of slave trees $M = 3$ is shown in Fig. 5. The candidate trees are highlighted by rectangular boxes with rounded corners, while the slave trees are highlighted by rectangular boxes. Again, solid, dashed, and dotted lines are used to distinguish candidate and slave trees with depths 0, 1, and 2, respectively. The induction is described as follows.

1) Build four primitive PTs T_0, T_1, T_2, and T_3, with the Jaccard similarities calculated from (1) being 0.6585, 0.5217, 0.4545, and 0.4348, respectively.
2) Assign T_0 and T_1 that have the highest similarities to class term X as the candidate trees C_{00} and C_{01}, with the first subscript referring to the depth of the candidate tree and the second referring to the index of the candidate tree (from 0 to $L - 1$).
3) Now consider the tree aggregation from depth 0 to depth 1. For the first candidate tree $C_{00} = T_0$, try the rest of the primitive PTs T_1, T_2, and T_3 in turn to aggregate using different aggregation operators MIN, MAX, OWA, and WA. The best aggregated PTs are T_4, T_5, and T_6, with similarities being 0.6585, 0.7778, and 0.7021. Note that there is no nonprimitive slave trees available to grow the candidate tree.
4) For the second candidate tree $C_{01} = T_1$, try the rest of the primitive PTs T_0, T_2, and T_3 in turn to aggregate using different aggregation operators MIN, MAX, OWA, and WA. The best aggregated PTs are T_7, T_8, and T_9, with similarities being 0.6585, 0.7814, and 0.7128. Again, no aggregations between the candidate tree and nonprimitive slave trees happen.
5) Assign T_8 and T_5 that have one depth and the highest similarities as the candidate trees C_{10} and C_{11}. Also assign T_8, T_5, and T_9 that have the highest similarities as the slave trees S_{10}, S_{11}, and S_{12}, with the first subscript referring to the depth of the slave tree and the second referring to the index of slave tree (from 0 to $M - 1$).
6) Now consider the tree aggregation from depth 1 to depth 2. For the first candidate tree $C_{10} = T_8$, use the primitive PTs T_0 and T_3 that do not appear in the candidate tree to aggregate with. The aggregated trees are T_{10} and T_{11}, with similarities being 0.7772 and 0.7814, respectively. Then use the slave trees T_5 and T_9 to aggregate with C_{10}, resulting in T_{12} and T_{13}, with similarities being 0.7962 and 0.7814.
7) Similarly for the second candidate tree $C_{11} = T_5$ at depth 1, four aggregated trees T_{14}, T_{15}, T_{16}, and T_{17} are generated, with similarities being 0.7792, 0.7778, 0.7962, and 0.7788, respectively.
8) This induction is carried out in the same manner from depth 2 to depth 3. T_{18}, T_{19}, T_{20}, and T_{21} are constructed. As none of them has a higher similarity than $C_{20} = T_{12}$, the building process stops and C_{20} is returned.

It is worth noting that the duplication removal is performed in selecting the candidate trees and slave trees. For example, at depth 2, although T_{16} has a high similarity measure, it is not selected as a candidate tree or a slave tree since tree T_{12} is identical to it, and also T_{12} is constructed earlier.

It can be shown from Figs. 4 and 5 that the search space of a SPT is subsumed by that of a general PT, as every PT considered in the SPT induction has been considered in the general tree induction. Generally, the greater the sizes of candidate (L) and slave trees (M), the larger search space the induction of an optimal general PT can reach, and thus, less likely the search is to be trapped in a local optimum. For example, PT T_7 in Fig. 4 is the optimum tree from SPT induction; however, it is only a local optimum tree (corresponding to T_{14}) in Fig. 5 in general PT induction. This tree is not chosen to proceed as its peers T_{12} and T_{11} have higher similarities. In this paper, the setting of $L = 2$ and $M = 3$ is used throughout all examples and experiments to tradeoff the search effort and the capacity of escaping from local optima. On the other hand, the configuration of $L = 1$ and $M = 0$ corresponds to the induction of simple PTs [4], which

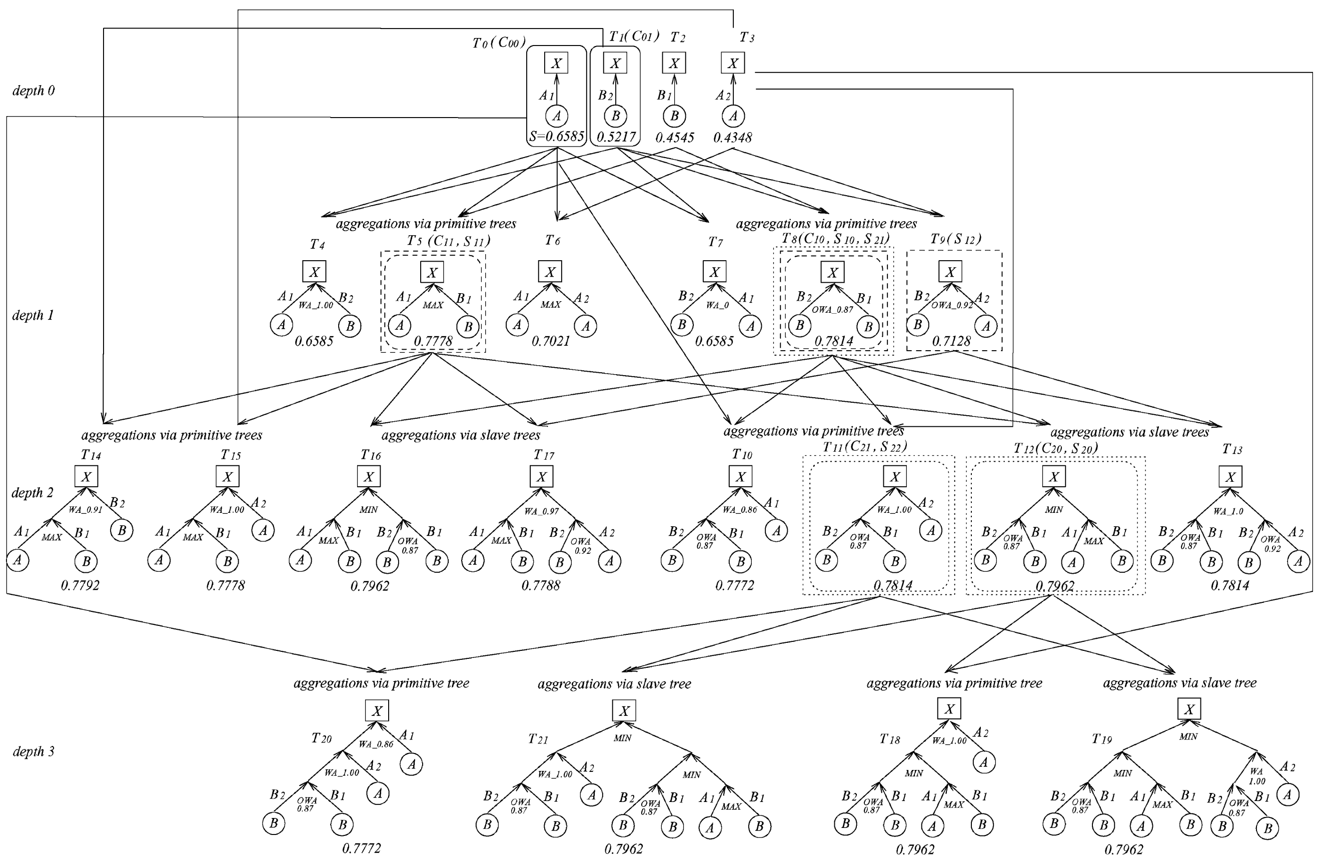

depth 0
depth 1
depth 2
depth 3
aggregations via primitive trees
aggregations via slave trees
aggregations via primitive tree
aggregations via slave tree
S=0.6585
0.5217
0.4545
0.4348
0.7778
0.7814
0.7962

Fig. 5. Induction of a general PT.

attempts to find optimal simple PTs with little computation effort.

IV. Related Work

Related work includes the fuzzy SBM [3], its extension, the WSBM [12], and the well-known FDT induction [5], [20]. In fact, the first two are two specific cases of PT induction. All three methods are briefly reviewed, and they, along with the PT induction, are evaluated using the small dataset given in Table II.

A. SBM

The SBM consists of three main steps:

1) classifying training data into subgroups according to class values, with each group having the data that prefer voting for one class;
2) calculating *fuzzy subsethood* values of a certain output class to each input fuzzy term. A fuzzy subsethood value of X with regard to $A_1, \mathrm{Sub}(X, A_1) = (X \cap A_1)/X$, represents the degree to which X is a subset of A_1 (see [3] and [12]), where $\cap$ is a t-norm operator (MIN is used);
3) creating rules based on fuzzy subsethood values. In particular, the fuzzy term for each input variable that has the highest subsethood value, and such value is greater than or equal to a prespecified threshold value $\alpha \in [0, 1]$ (0.9 used in [3]), is selected. The selected fuzzy terms for all input variables are used to construct a fuzzy rule using t-norm MIN.

Regardless of the subgrouping process and the use of *subsethood* rather than *similarity* measure, the SBM always generates a PT that has only one depth. In this tree, the fuzzy terms that have the greatest subsethood values per input variable and whose values are greater than or equal to α are aggregated via a t-norm operator (MIN) to predict a class concerned.

For the dataset in Table II, two groups are created. The first has the first four data points, which prefer voting for class X rather than Y, and the second has the remaining two data points. Assume that class X is considered. The subsethood values of X to $A_i, B_i, i = 1, 2$, are calculated as 0.6970, 0.4848, 0.4848, and 0.6061, respectively. If α is set to 0.9 as in [3], no fuzzy rules (PTs) can be generated: although fuzzy terms A_1 and B_2 have the highest subsethood values among input variables A and B, respectively, their values are not greater than the α value. They are thus not qualified to construct a fuzzy rule (PT). However, for comparison purposes, α is set to 0.6 to generate the PT for class X, as shown in Fig. 6. This figure also shows the PT for Y that is constructed in the same manner.

B. WSBM

The WSBM is a weighted version of the SBM. It uses a certain weighting strategy to represent fuzzy terms rather than choosing the one that has the greatest subsethood value per variable. In

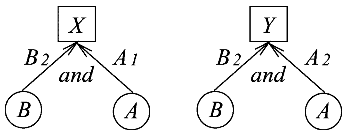


Fig. 6. PTs generated by SBM.

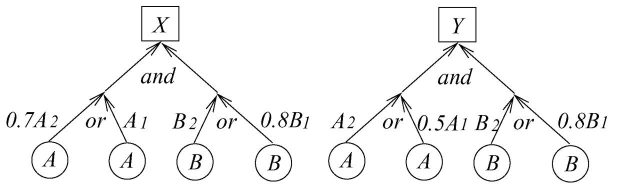


Fig. 7. PTs generated by WSBM.

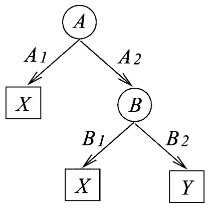


Fig. 8. Decision tree generated by FDT induction.

particular, the weight for fuzzy term A_i is defined as

$$w(X, A_i) = \frac{\mathrm{Sub}(X, A_i)}{\max_{j=1,2}\mathrm{Sub}(X, A_j)}. \tag{11}$$

Using this equation, the weights for A_1, A_2, B_1, and B_2 in the classification of X are calculated as 1.0, 0.7, 0.8, and 1.0, respectively. The WSBM then constructs a PT with two depths. On the bottom depth, all fuzzy terms (with different weights) for each variable aggregate via a t-conorm operator, on the top depth the aggregated values, further aggregate via a t-norm operator. The trees, generated by WSBM using data in Table II, are shown in Fig. 7.

Unlike the proposed PT induction that can generate patter trees with various depths, the SBM and WSBM generate trees with fixed depths (one and two, respectively). The limited depths of the SBM and WSBM trees may prevent them from obtaining high accuracy in classification applications (see Section IV-E).

C. FDT Induction

Inspired by the classic decision tree induction by Quinlan [10], there is substantial work on FDTs. For example, Yuan and Shaw [20] have used fuzzy entropy to generate FDTs. Janikow [5], Olaru and Wehenkel [8] have presented different FDT inductions. An FDT as shown in Fig. 8 can be built using [20] over the dataset in Table II. It has a root on the top and several leaf nodes at the bottom. When a new data sample needs to be classified by the FDT, it traverses from the top to the bottom. The output class of a leaf node in the bottom, which the data sample reaches with the highest truth value, is chosen as the prediction class.

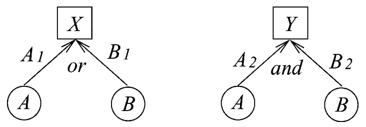


Fig. 9. Alternative PTs converted from the decision tree in Fig. 8.

For the same training dataset, FDT induction may generate different results with a different minimal number of instances per leaf, which is used as a criterion to terminate the tree building process. Considering that only six data samples are available, this criterion can be set to 1–6. Among all these, the best result (as shown in Fig. 8, when the minimal number of instances per leaf is set to 1 or 2) has been chosen for comparison in Section IV-E.

D. Relation Between FDTS and Pattern Trees

An FDT can be converted to a set of PTs whose size is equal to the number of output classes. The easiest way is to convert an FDT to a set of semibinary PTs. That is, there are maximally two branches allowed for each node in the PT except for the root. The conversion is outlined as follows. For each fuzzy rule (a branch from the root to a leaf node) in a decision tree, the input fuzzy terms are connected by a t-norm operator (usually MIN) in different depths of the PT. A fuzzy rule consisting of n fuzzy terms results in an $(n-1)$-depth binary PT as the bottom depth contains two fuzzy terms. The fuzzy rules that have the same classification class are connected by a t-conorm (usually MAX) at the top depth (depth n) to construct a PT for this output class. The number of depths of the generated semibinary PTs remains the same as the FDT, regardless of whether the decision tree is binary or not. Fig. 3 shows the PTs that are equivalent to the decision tree, as shown in Fig. 8. They both have the same rule base as listed in (8)–(10). Note that the conversion from a decision tree to PTs is not unique. For example, an alternative conversion from Fig. 8 is shown in Fig. 9, which can be represented by two rules as follows:

$$\text{Rule 1 : IF } A = A_1 \text{ OR } B = B_1 \text{ THEN class } = X \tag{12}$$

$$\text{Rule 2 : IF } A = A_2 \text{ AND } B = B_2 \text{ THEN class } = Y. \tag{13}$$

These two fuzzy rules are functionally equal to rules (8)–(10). When the size of fuzzy terms for each variable increases, the conversion of a FDT to multibranch PTs becomes complicated and the work on that is on-going.

It is worth stressing that decision trees and PTs are different in terms of four aspects.

1) The former focus on separating data samples that have *different* output classes, while the latter focus on representing the structures of data samples that have the *same* output classes.

2) For each internal node, the former consider *all* fuzzy terms of the chosen input variable in the downward branches, while the latter only consider *one* in the upward branch.

3) The former normally make use of MIN and MAX aggregations *only*, while the latter can use *any* aggregations, as described in Section II-B.

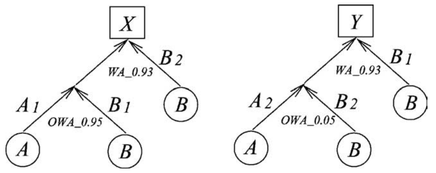


Fig. 10. PTs generated using SPT induction.

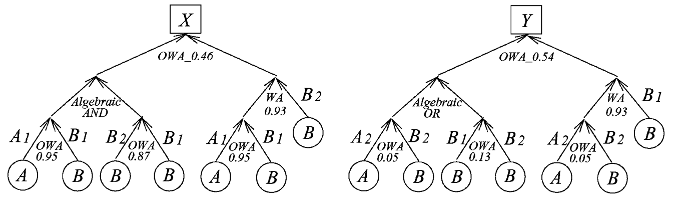


Fig. 11. PTs generated using general PT induction.

TABLE VI
RESULTS OF SBM, WSBM, FDT, AND PTs

	No.	RMSE(X)	RMSE(Y)	ARMSE
SBM	4	0.3916	0.2000	0.2958
WSBM	4	0.2386	0.3435	0.2911
FDT	5	0.2000	0.2000	0.2000
SPT	6	0.1956	0.1956	0.1956
PT	6	0.1715	0.1715	0.1715

4) The tree induction methods are completely different, with the former based on the heuristics of *entropy measure* [20] while the latter is based on the heuristics of *similarity measure*.

E. Comparison Using the Small Dataset

This section facilitates the comparison between SBM, WSBM, FDTs, SPTs (with $L = 1$ and $M = 0$) and PTs (with $L = 2$ and $M = 3$) over the small dataset given in Table II. All aggregations, including MIN/MAX, algebraic AND/OR, Łukasiewicz, EINSTEIN, OWA, and WA are considered in SPT and PT inductions, and both Jaccard and RMSE-based similarities are tried, and the latter is reported, as it produces better results. The maximum depth d is set to 3 for this toy example. Figs. 10 and 11 show the PTs generated by SPT induction and general PT induction, respectively. It is worth noting that, if only MIN/MAX are allowed in the SPT induction, the generated trees are exactly the same as those shown in Fig. 9, with AND being MIN and OR being MAX.

Table VI shows the classification results of the SBM, WSBM, FDT, SPT, and PT over the dataset in Table II. The comparison includes the number of correctly classified data samples (No.), the RMSE of the classification for classes X, Y, and their average (ARMSE). It is clear that the PTs perform the best, in terms of both the correctly classified number and the mean square error, among all the methods. SPTs perform slightly worse, but still better than the FDT. This example shows that the SBM and WSBM cannot get good results, due to the fact that their fixed depths prevent them from finding optimal tree structures. FDTs, on the other hand, ignore some candidate trees involved with t-conorm aggregations. For instance, in this example, Fig. 8 considers whether A_1, A_2, $A_1 \vee A_2$, and $A_2 \wedge B_1$, etc., are important or not for the classification, but not $A_1 \vee B_1$ explicitly, thus failing to find such an important rule. In contrast, the proposed PT induction method explicitly makes use of both t-norm and t-conorm aggregations of fuzzy terms in building trees. The induction provides a ground on which the t-norm and t-conorm compete with each other to see which can provide more optimal trees. In this example, PT induction evaluates not only A_1, $A_2, A_1 \vee A_2$, and $A_2 \wedge B_1$, but also $A_1 \vee B_1$. It is thus more likely to find optimal trees.

TABLE VII
CLASSIFICATION ACCURACY OF SBM, WSBM, AND FOUR VARIANTS OF PTs USING IRIS PLANT DATASET

	SBM	WSBM	Four variants
Exp1	80.00%	93.33%	94.66%
Exp2	78.67%	93.33%	97.33%

TABLE VIII
EXPERIMENTAL DATASETS SUMMARY

Name	Number of data	Number of inputs	Number of classes
Iris-Plant	150	4	3
Wisconsin Breast Cancer	699	9	2
Glass identification	214	9	6
Diabetes	768	8	2
Wine recognition	178	13	3
Credit	690	15	2

V. EXPERIMENTAL RESULTS

To demonstrate the effectiveness of the proposed PT induction method, Section V-A presents comparative studies with the fuzzy SBM [3] and the WSBM [12] using the Iris Plant dataset. In addition, Section V-B presents an extensive comparison to the C4.5 decision tree (C4.5) [10], nearest neighbor (NN) [1], J. Platt's sequential minimal-optimization-based support vector classifier (SVM) [9], and FDTs [20], over the datasets obtained from the University of California, Irvine (UCI) machine learning repository [2], which include Iris Plant, Wisconsin Breast Cancer, Glass Identification, Diabetes, Wine Recognition, and Credit.

Four variants of PTs participate in the comparison: SPTs with $L = 1, M = 0$, and $d = 5$ (SPT5), SPTs with $L = 1, M = 0$, and $d = 10$ (SPT10), general PTs with $L = 2, M = 3$, and $d = 5$ (PT5), and general PTs with $L = 2, M = 3$, and $d = 10$ (PT10). Note that both SPTs and general PTs may not have the depths as specified, as the tree induction stops once the highest similarity does not increase (as described in the *break* statements in Algorithms 1 and 2). These four variants make use of RMSE-based similarity and all aggregations listed in Section II-B. It is worth noting that the use of aggregations rather than MIN and MAX inevitably deteriorates the transparency of trees. However, the PTs can have significant performance gain, thanks to the variety of aggregations, especially WA and OWA.

TABLE IX
TENFOLD CROSS-VALIDATION CLASSIFICATION ACCURACY AND STANDARD DEVIATION OF C4.5, SVM, KNN, FDT, AND FOUR VARIANTS OF PT OVER SIX DATASETS

Data Set	C4.5	SVM	KNN	FDT	SPT5	SPT10	PT5	PT10
Iris Plant	96.00±5.62	96.00±4.66	95.33±5.49	94.67±5.26	97.33±3.44	97.33±3.44	96.67±4.71	96.67±4.71
Wisconsin Breast Cancer	94.56±3.63	96.99±2.07○	94.99±3.71	94.71±2.24	96.28±2.45	95.99±2.52	96.99±2.19○	96.56±2.36
Glass Identification	66.75±7.94	56.13±7.27●	70.50±7.27	64.03±10.17	61.67±4.31	66.34±3.04	62.62±6.03	65.39±4.09
Diabetes	73.83±5.66	77.34±4.07○	70.17±4.69	72.91±4.46	73.84±5.42	73.71±5.73	74.74±6.30	73.83±6.53
Wine Recognition	93.86±5.52	98.33±2.68○	94.97±4.11	92.71±5.87	94.90±4.22	93.24±5.79	94.41±5.86	93.82±6.11
Credit	86.09±3.75	84.93±4.39	81.16±4.83●	85.36±4.45	85.94±4.83	85.94±5.11	85.51±4.32	85.94±4.83

○, ● statistically significant improvement or degradation

A. *Classification Accuracy Comparison With SBM and WSBM*

Iris Plant dataset is a classic small benchmark for machine learning algorithm evaluation. It has 150 data samples, with each having four input variables and one of three class labels. The configuration is set up according to [12]: 1) a simple fuzzification method based on three evenly distributed trapezoidal membership functions for each input variable is used to transform the crisp values into fuzzy values and 2) the whole data are divided into two parts, namely the odd-labeled data and the even-labeled data. Two experiments are carried out, with the first (Exp1) using odd-labeled data as the training set and even-labeled data as the test set. The second (Exp2) uses even-labeled data as the training set and odd-labeled data as the test set. Table VII shows results of different methods. As can be seen, the four variants of PTs[3] produce better results than do the SBM and WSBM.

B. *Classification Accuracy Comparison With C4.5, KNN, SVM, and FDT*

The C4.5, KNN, and SVM classifiers implemented in the Weka machine learning toolkit [17] form a crisp classifier group. The default settings of each classifier in the toolkit are used. For example, the minimal number of instances per leaf is set to 2 for C4.5 and the number of neighbors to use is set to 1 for KNN. The FDT and four variants of PTs SPT5, SPT10, PT5, and PT10 form a fuzzy classifier group. After trials of experiments, the minimal number of instances per leaf for FDT is tuned to 10 in the experiments. On the other hand, the four variants of PTs do not have any parameters to tune. For the classifiers in a fuzzy group, a simple fuzzification method based on six evenly distributed trapezoidal membership functions for each input variable is used to transform the crisp values into fuzzy values. The experiments presented in this section compare the classification accuracy of different classifiers in their original prototypes. There is no attempt made to any optimizations, although the optimizations do exist, such as the tree pruning for C4.5 and FDT, and the fuzzification optimization for FDT and variants of PTs.

The datasets of Iris Plant, Wisconsin Breast Cancer, Glass Identification, Diabetes, Wine Recognition, and Credit obtained from the UCI machine learning repository [2], which have been widely used as benchmarks in classification applications, are summarized in Table VIII. Each dataset is stratified and divided into tenfold of (approximately) equal size. Each time one fold is left out of the whole dataset from training, and this fold is used for testing. As a result, there are ten runs for each classifier based on one dataset. Table IX shows the average classification accuracy and the corresponding standard deviation of all ten runs per dataset and classifier. Fig. 12 shows the average, minimal, and maximal classification accuracy among all ten runs per dataset and classifier. The paired student's t-test is employed to justify if a classifier is statistically better than another for a given dataset. The ○ and ● in Table IX indicate statistically significant improvement or degradation (with a confidence level of 95%) of other classifiers, compared to the baseline C4.5 classifier. That means, for example, the SVM has a significant win and loss compared to C4.5 over Wisconsin Breast Cancer and Glass Identification datasets, respectively. Similarly, the paired student's t-test can be applied to any pair of classifiers for a given dataset. Table X summarizes the wins, losses, and ties of classifiers among all the tests. The first column in Table X is the difference between the number of wins and the number of losses that is used to generate the ranking [17]. As can be seen, the SVM, PT10, and PT5 are tied with the net wins, although they have different individual wins and losses. The performance of the other two variants of PT is not as good, but they are still better than the FDT, C4.5, and KNN.

C. *Overfitting Comparison With C4.5, KNN, SVM, and FDT*

Overfitting refers to the phenomenon that a classifier may fit well to the training data but is not generalized enough to classify unseen data. The tenfold cross-validation-based experiments fairly present the normal behaviors of classifiers but they do not reveal which classifiers are prone to overfitting. In this section, the whole data (rather than the tenfold cross-validation data) are used to train and test C4.5, KNN, SVM, FDT, and four variants of the PT using the same training parameters as presented in Section V-B. The results are collectively shown in Table XI and also in Fig. 12. For each classifier, the RMSE of the classification accuracies for six datasets between the tenfold cross-validation and whole data-based is shown at the bottom of the table. The RMSE reveals how much improvement one classifier can gain based on the whole data experiment, compared to the tenfold cross-validation one in average for six datasets. It is assumed that the more gain for one classifier (the longer the distance between the dashed lines and the solid lines in Fig. 12), the more likely that the classifier is prone to overfitting. In other words, the close performance between the tenfold cross-validation and whole data-based experiments would indicate the robustness to overfitting. The results of this test reveal that the SVM maintains the best generality (with RMSE being

[3]They produce the same results.

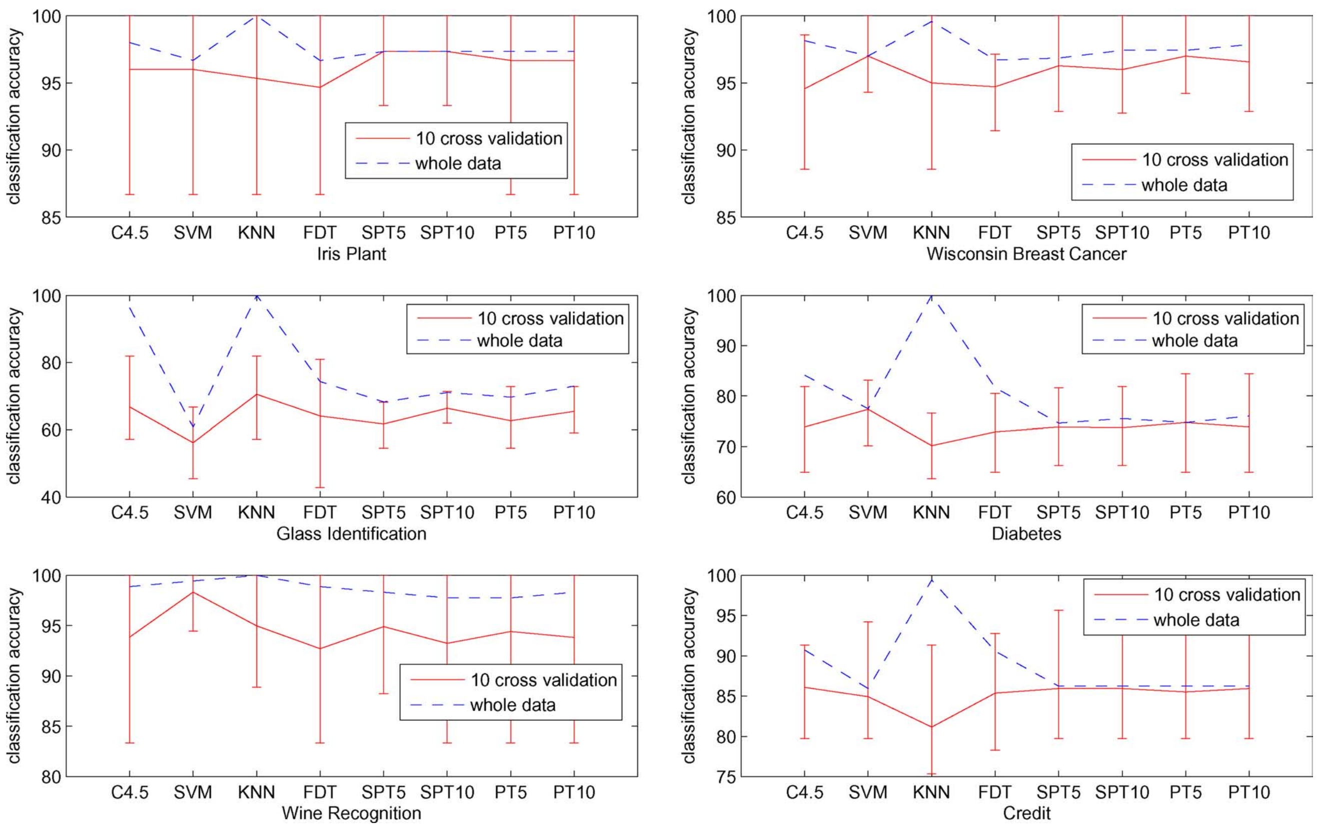


Fig. 12. Average, minimum, and maximum classification accuracy of C4.5, SVM, KNN, FDT, and four variants of PT over six datasets.

TABLE X
RANKING OF C4.5, SVM, KNN, FDT, AND FOUR VARIANTS OF PT BASED ON SIX DATASETS

Result set	Wins−Losses	Wins	Losses	Ties
SVM	3	8	5	29
PT10	3	4	1	37
PT5	3	4	1	37
SPT10	2	3	1	38
SPT5	-1	2	3	37
FDT	-2	2	4	36
C4.5	-2	2	4	36
KNN	-6	4	10	28

2.00) and four variants of PT trees perform slightly worse (with RMSE being 3.04, 2.82, 3.20, and 3.73, respectively). The FDT performs worse than the SVM and variants of PT, but better than C4.5. It is not surprising that KNN performs the worst, as it is totally biased to the nearest neighbor in classification and makes no attempt to find a general model.

For the tree-based classifiers, the reason that C4.5 and the FDT are more prone to overfitting than variants of PT is that the former endeavor to find a set of rules (branches), with each representing a portion of the training data. It is expected that the cooperation of the whole rules using a t-conorm leads to a good classification model. Unfortunately, the construction of each rule is based on a portion of training data, which causes the overfitting of individual rules and, in turn, causes the overfitting of the whole model. This problem gets worse when a lengthy rule is constructed, as such a rule represents only a small amount of training data rather than a meaningful pattern. In contrast, the latter uses the whole training data to find a rule (tree) for each class. The growth of the tree is permitted only if all training data (in contrast to a portion of training data) can be better fit into the tree. Therefore, even complex PTs do not suffer from overfitting. In other words, the decision tree seeks various local optimal structures that fit various portions of training data, while the PT seeks a global tree structure to represent the whole training data.

D. Structure Complexity

SPTs have compact structures. Each SPT5 and SPT10 has 6 and 11 leaf nodes, respectively. Fig. 13 shows three SPT5 trees using the Wine dataset (with classification accuracy of 94.44%) in the first of tenfold cross-validation runs. The ellipses are the input variables and the rectangles are the output classes (0, 1, or 2). Fi's, $i = 0, \ldots, 5$, are the fuzzy terms associated with each input variable. PT5 and PT10 may have complicated tree structures because each PT5 may have up to $2^5 = 32$ leaf nodes and each PT10 may have up to $2^{10} = 1024$ leaf nodes. However, they usually do not have these maximums, as the trees are not fully spanned. Fig. 14 shows a PT5 tree for class 0 using the Wine Recognition dataset in one of tenfold cross-validation runs. Such a tree has only nine leaf nodes, much less than the maximum of 32. For a rough comparison, the average number of leaf nodes of FDTs constructed using the Wine Recognition dataset in the tenfold cross-validation runs is 59. This reveals that PTs, especially SPT5 and SPT10, can have more compact

TABLE XI
WHOLE DATA-BASED CLASSIFICATION ACCURACY OF C4.5, SVM, KNN, FDT, AND FOUR VARIANTS OF PT OVER SIX DATASETS

Data Set	C4.5	SVM	KNN	FDT	SPT5	SPT10	PT5	PT10
Iris Plant	98.00	96.67	100.00	96.66	97.33	97.33	97.33	97.33
Wisconsin Breast Cancer	98.14	97.00	99.57	96.70	96.85	97.42	97.42	97.85
Glass Identification	96.26	60.74	100.00	74.29	68.22	71.03	69.63	72.90
Diabetes	84.11	77.47	100.00	81.64	74.61	75.52	74.74	76.04
Wine Recognition	98.88	99.44	100.00	98.87	98.31	97.75	97.75	98.31
Credit	90.72	85.94	99.42	90.57	86.23	86.23	86.23	86.23
RMSE	13.16	2.00	18.98	8.24	3.04	2.82	3.20	3.73

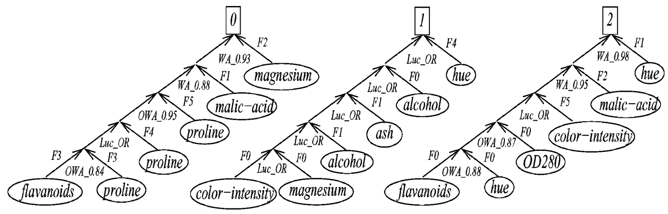


Fig. 13. SPTs generated using Wine Recognition dataset.

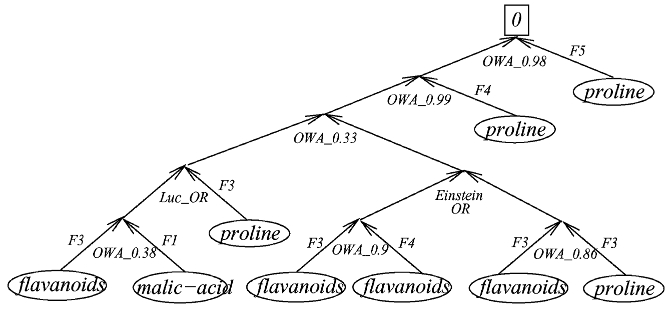


Fig. 14. PT generated using Wine Recognition dataset for class 0.

tree structures than FDTs. However, it can be argued that FDTs may be simplified by tree pruning.

In general, SPTs not only produce high classification accuracy, but also preserve compact tree structures, while general PTs can produce even better accuracy, but as a compromise, produce more complex tree structures. With respect to choosing different variants of PTs induction methods for real-world applications, PT10 are favored if performance is a critical factor. If, however, the comprehensibility is critical for the solution to be considered, SPT5 or SPT10 trees are a good choice.

VI. CONCLUSION

This paper proposed a type of tree termed *PTs* that makes use of different aggregations, including both t-norms and t-conorms. Like decision trees, PTs are an effective tool for classification applications. This paper discussed the difference between decision trees and PTs, and also showed that the SBM and the WSBM are two specific cases of PT induction.

A novel PT induction method was proposed using similarity measure and fuzzy aggregations. The comparison to other classification methods including SBM, WSBM, C4.5, nearest neighbor, SVM, and FDT induction showed that: 1) PTs can obtain high accuracy rates in classifications; 2) PTs are robust to overfitting; and 3) PTs, especially simple PTs, maintain compact tree structures.

Although the proposed PT induction method shows promising results, it does not mean that it cannot be improved. Other heuristics rather than the similarity measure can be used to guide the PTs induction. In addition, the underlying relation between decision trees and PTs needs more research work. The conversion between decision trees and PTs is worth investigating.

ACKNOWLEDGMENT

The authors would like to thank Prof. L. Kóczy, S. U. Mendis, J. Botzheim, and D. Wilkinson for their helpful discussions on the work presented here. In addition, the authors are very grateful for the anonymous referees for their helpful comments.

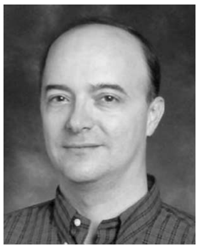
REFERENCES

[1] D. Aha and D. Kibler, "Instance-based learning algorithms," *Mach. Learn.*, vol. 6, no. 1, pp. 37–66, 1991.
[2] A. Asuncion and D. J. Newman. (2007). UCI machine learning repository, [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html
[3] S. M. Chen, S. H. Lee, and C. H. Lee, "A new method for generating fuzzy rules from numerical data for handling classification problems," *Appl. Artif. Intell.*, vol. 15, no. 7, pp. 645–664, 2001.
[4] Z. H. Huang and T. D. Gedeon, "Pattern trees," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, 2006, pp. 1784–1791.
[5] C. Z. Janikow, "Fuzzy decision trees: Issues and methods," *IEEE Trans. Syst., Man, Cybern.*, vol. 28, no. 1, pp. 1–14, Feb. 1998.
[6] L. T. Kóczy, T. Vámos, and G. Biró, "Fuzzy signatures," in *Proc. Int. Conf. Soft Intell. Comput.*, 1999, pp. 210–217.
[7] M. Nikravesh, "Soft computing for perception-based decision processing and analysis: Web-based BISC-DSS," in *Studies in Fuzziness and Soft Computing,* vol. 164. Berlin/Heidelberg, Germany: Springer-Verlag, 2005, pp. 93–188.
[8] C. Olaru and L. Wehenkel, "A complete fuzzy decision tree technique," *Fuzzy Sets Syst.*, vol. 138, no. 2, pp. 221–254, 2003.
[9] J. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods—Support Vector Learning*. Cambridge, MA: MIT Press, 1999, pp. 185–208.
[10] J. R. Quinlan, "Decision trees and decision making," *IEEE Trans. Syst., Man, Cybern.*, vol. 20, no. 2, pp. 339–346, Mar./Apr. 1990.
[11] G. V. S. Raju and R. A. Kisner, "Hierarchical fuzzy control," *Int. J. Control*, vol. 54, no. 5, pp. 1201–1216, 1991.
[12] K. A. Rasmani and Q. Shen, "Weighted linguistic modelling based on fuzzy subsethood values," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, 2003, vol. 1, pp. 714–719.
[13] B. Schweizer and A. Sklar, "Associative functions and abstract semigroups," *Publicationes Mathematicae Debrecen*, vol. 10, pp. 69–81, 1963.
[14] A. Suárez and J. F. Lutsko, "Globally optimal fuzzy decision trees for classification and regression," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 12, pp. 1297–1311, Dec. 1999.

[15] L. X. Wang and J. M. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, no. 6, pp. 1414–1427, Nov./Dec. 1992.
[16] X. Wang, B. Chen, G. Olan, and F. Ye, "On the optimization of fuzzy decision trees," *Fuzzy Sets Syst.*, vol. 112, no. 1, pp. 117–125, 2000.
[17] I. H. Witten and E. Frank, *"Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. San Mateo, CA: Morgan Kaufmann, 2005.
[18] K. W. Wong, T. D. Gedeon, and L. T. Kóczy, "Construction of fuzzy signature from data: An example of SARS pre-clinical diagnosis system," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, 2004, vol. 3, pp. 1649–1654.
[19] R. R. Yager, "On ordered weighted averaging aggregation operators in multicriteria decision making," *IEEE Trans. Syst., Man, Cybern.*, vol. 18, no. 1, pp. 183–190, Jan./Feb. 1988.
[20] Y. Yuan and M. J. Shaw, "Induction of fuzzy decision trees," *Fuzzy Sets Syst.*, vol. 69, no. 2, pp. 125–139, 1995.
[21] L. A. Zadeh, "Fuzzy sets," *Inf. Control*, vol. 8, no. 3, pp. 338–353, 1965.

Zhiheng Huang received the B.Sc. degree in industrial equipment and control engineering and computer science from South China University of Technology, Guangzhou, China in 2000, and the M.Sc. and Ph.D. degrees in artificial intelligence from the University of Edinburgh, Edinburgh, U.K., in 2001 and 2006, respectively.

Since May 2006, he has been a Postdoctoral Researcher at Berkeley Initiative in Soft Computing (BISC), University of California, Berkeley. His current research interests include machine learning, data mining, intelligent data analysis, fuzzy set theory, information retrieval, information extraction, and question–answer systems.



Tomás D. Gedeon received the B.Sc. and Ph.D. degrees in computer science from the University of Western Australia, Perth, Australia, in 1981 and 1989, respectively.

He is currently the Chair Professor in the Department of Computer Science, Australian National University, Canberra, A.C.T., where he was the Associate Director of Education at the College of Engineering and Computer Science, and was also a former Head of the Department of Computer Science. He was formerly at Murdoch University and the University of New South Wales. He also serves on a number of journal advisory boards as a Member, an Associate Editor, or an Editor. His current research interests include the development of automated systems for information extraction, and for the synthesis of the extracted information into humanly useful information resources (hierarchical knowledge), mostly using fuzzy systems and neural networks. He is also interested in cognitive modeling based on biologically plausible information flow constraints, and in eye gaze and face recognition.

Prof. Gedeon has been nominated three times for VC's awards for postgraduate supervision. He was the President of the Asia-Pacific Neural Network Assembly.

Masoud Nikravesh received the B.Sc. degree in chemical engineering from the Abadan Institute of Technology, Abadan, Iran, in 1985, and the M.Sc. and Ph.D. degrees in chemical engineering from the University of South Carolina, Columbia, in 1993 and 1994, respectively.

He is currently the CITRIS Executive Director, Computational Science and Engineering, University of California, Berkeley, where he was a Postdoctoral Researcher in the Materials Science and Mineral Engineering Department during 1995, and at Lawrence Berkeley National Laboratory, Earth Sciences Division, as a joint appointment. He is the author or coauthor of nine books and more than 200 papers on a wide range of scientific and engineering applications. In addition, he has been an Invited Lecturer throughout the world, including USA, China, Germany, Hong Kong, Finland, Canada, U.K., Turkey, New Zealand, and Mexico.

Dr. Nikravesh has been a member of the Society of Plastics Engineers (SPE), the American Association of Petroleum Geologists (AAPG), the Society of Exploration Geophysicists (SEG), the American Chemical Society (ACS), the North American Fuzzy Information Processing Society (NAFIPS), the International Frequency Sensor Association (IFSA), AICHE, and other scientific scholarly societies. He has also been an Invited Lecturer at many conferences and scientific events.