2006 IEEE International Conference on Fuzzy Systems
Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada
July 16-21, 2006

Pattern Trees

Zhiheng Huang† and Tamás D. Gedeon

Abstract— **This paper proposes a new type of tree termed *pattern trees*. Like decision trees, pattern trees are an effective tool for classification applications. This paper discusses the difference between decision trees and pattern trees, and also shows that the subsethood based method and the weighted subsethood based method are two specific cases of pattern trees.**

A novel pattern tree induction method is proposed. The comparison to other classification methods including fuzzy decision tree induction shows that pattern trees can obtain higher accuracy rates in classifications. In addition, pattern trees are capable of generating patterns with good generality, while decision trees can easily fall into the trap of over-fitting.

I. Introduction

The major advantage of using fuzzy rules for classification (prediction) applications is to maintain transparency as well as a high accuracy rate. The difficulty in obtaining an ideal classification model is to balance the complexity of a rule base and the accuracy. Generally, fuzzy rule models can produce arbitrary accuracy if unlimited rule numbers are allowed [10]. However, this inevitably violates the original motivation of using fuzzy rules – transparency. To address the balance, there are many fuzzy rule induction methods. In Wang and Mendel [9] an algorithm for generating fuzzy rules by learning from examples has been presented. Yuan and Shaw [13] have proposed the induction of fuzzy decision trees for generating fuzzy classification rules (the extension of the classic decision tree induction method by Quinlan [5]). Chen, Lee and Lee [2] have presented a subsethood based method (SBM) for generating fuzzy rules. Rasmani and Shen [7] have proposed a weighted fuzzy subsethood based rule induction method (WSBM). To avoid the exponential growth of the size of rule base when the number of input variables increases, Raju, Zhou and Kisner [6] have proposed hierarchical fuzzy systems. Recently, Kóczy, Vámos and Biró [3], and Wong, Gedeon and Kóczy [11] have presented fuzzy signatures which model the complex structure of the data points in a hierarchical manner.

Most of the existing fuzzy rule induction methods such as the fuzzy decision trees [13] focus on searching for the rules which only use t-norm operators [8] (e.g., the MIN or algebraic MIN). The disregarding of the t-conorms is due to the fact that any rule using t-conorms can be represented by several rules which only use t-norms. 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 clearly shown in an artificial dataset in Section IV. Recently, Kóczy, Vámos and Biró [3] have proposed fuzzy signatures to model the complex structures of the data points, in which different aggregation operators including MIN, MAX, and average etc. are used.

This paper proposes a new type of tree termed *pattern trees*. Like decision trees, pattern trees are an effective tool for classification applications. A novel pattern tree induction method is proposed to build the pattern trees, by means of the similarity measures and different aggregation operators. The experiments show that the pattern trees can obtain higher accuracy rates than the SBM, WSBM and the fuzzy decision tree induction in classifications. In addition, pattern trees perform more consistently than fuzzy decision trees. The former are capable of generating patterns with good generality, while the latter can easily fall into the trap of over-fitting.

The rest of the paper is arranged as follows: Section II provides the definitions for similarity, aggregations and pattern trees. Section III proposes a novel pattern tree induction method. Section IV shows that the SBM and WSBM are two specific cases of pattern trees. It also outlines the difference between decision trees and pattern trees. Section V presents the experimental results. Finally, Section VI concludes the paper and points out important further research work.

II. Similarity, Aggregations and Pattern Trees

A. Similarity

Let A and B be two fuzzy sets [14] defined on the universe of discourse U. The fuzzy similarity [1] between them can be defined as:

$$S(A,B) = \frac{A \sqcap B}{A \sqcup B}, \tag{1}$$

where $\sqcap$ and $\sqcup$ denote a certain t-norm operator and a t-conorm respectively. Usually, the MIN ($\wedge$) and MAX ($\vee$) operators are used. According to the definition, $0 \leq S(A,B) \leq 1$. In practice, this measurement can be computed as

$$S(A,B) = \frac{\sum_{j=1}^{m}[\mu_A(x_j) \wedge \mu_B(x_j)]}{\sum_{j=1}^{m}[\mu_A(x_j) \vee \mu_B(x_j)]}, \tag{2}$$

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 pattern tree construction in Section III. Consider that the root mean square error (RMSE) of fuzzy sets A and B can be computed as

$$RMSE(A,B) = \sqrt{\frac{\sum_{j=1}^{m}(\mu_A(x_j) - \mu_B(x_j))^2}{m}}, \tag{3}$$

the RMSE based fuzzy set similarity can thus be defined as

$$S(A,B) = 1 - RMSE(A,B). \tag{4}$$

Zhiheng Huang is with the Department of Computer Science, the Australian National University, Canberra, ACT 0200, Australia (phone: 61-2-61258179; email: zhiheng.huang@anu.edu.au).

Tamás D. Gedeon is with the Department of Computer Science, the Australian National University, Canberra, ACT 0200, Australia (phone: 61-2-61251052; fax: 61-2-61250010; email: tom@cs.anu.edu.au).

0-7803-9489-5/06/$20.00/©2006 IEEE

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 weighted averaging (OWA) [12].

Triangular norms were introduced by Schweizer and Sklar [8] to model the distances in probabilistic metric spaces. In fuzzy sets theory, triangular norms (t-norm) and triangular conorms (t-conorm) are extensively used to model logical operators *and* and *or*. The basic t-norm and t-conorm pairs which operate on two fuzzy membership values a and b, $a, b \in [0, 1]$ are shown in Table I.

TABLE I
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	$\frac{ab}{2-(a+b-ab)}$	$\frac{a+b}{1+ab}$

Although the aggregations above are only shown to apply to a pair of fuzzy values, they can be applied to multiple fuzzy values as they retain associativity.

Definition 1: A WA operator of dimension n is a mapping $E : R^n \to R$, that has an associated n-elements 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

$$E(a_1, \ldots, a_n) = \sum_{j=1}^{n} w_j a_j.$$

Definition 2: An OWA operator [12] of dimension n is a mapping $F : R^n \to R$, that has an associated n-elements 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

$$F(a_1, \ldots, a_n) = \sum_{j=1}^{n} w_j b_j,$$

where b_j is the jth largest element of the collection $\{a_1, \ldots, a_n\}$.

A fundamental difference of the OWA from WA aggregation is that the former does not have a particular weight w_i associated for an element, rather a weight is associated with a particular ordered position of the element.

Example 1: Assume $w = (0.2, 0.3, 0.1, 0.4)^T$, then the WA operator on the vector of $(0.6, 1, 0.3, 0.5)$ is

$$E(0.6, 1, 0.3, 0.5) = 0.2 \times 0.6 + 0.3 \times 1 + 0.1 \times 0.3 + 0.4 \times 0.5 = 0.65,$$

and the OWA operator on the vector is

$$F(0.6, 1, 0.3, 0.5) = 0.2 \times 1 + 0.3 \times 0.6 + 0.1 \times 0.5 + 0.4 \times 0.3 = 0.55.$$

It is worth noting two special OWA operators are equal to MAX and MIN:

- If $w^* = (1, 0, \ldots, 0)$, then
$$F(a_1, \ldots, a_n) = max\{a_1, \ldots, a_n\}.$$
- If $w_* = (0, 0, \ldots, 1)$, then
$$F(a_1, \ldots, a_n) = min\{a_1, \ldots, a_n\}.$$

The main factor to determine which aggregation should be used relies on 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. 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 much of nonzero weights near the top will be more *or* than *and*.

C. Pattern Trees

A pattern tree is a tree which represents the pattern for an output class. The output class is located at the top as the root of this tree. The fuzzy terms of input variables are on different levels (except the top) of the tree. They use fuzzy aggregations (as presented in subsection II-B) to aggregate from the bottom level to the top (root).

For a classification application which involves several output classes, the worked model should have as many pattern trees as the number of the output classes, with each pattern tree representing one class. Assume two fuzzy variables A and B each have two fuzzy linguistic terms A_i and B_i, $i = \{1, 2\}$, and the task is to classify the data samples to either class X or Y. Fig. 1 shows two example pattern trees for classes X and Y. Note that all the nodes within the pattern trees are leaf nodes. When a new data sample is tested over a pattern tree, it starts from the bottom leaves and travels to the top. It finishes with a truth value indicating the degree of this data sample belonging to the output class of this pattern tree. The output class with the maximal truth value is chosen as the prediction class.

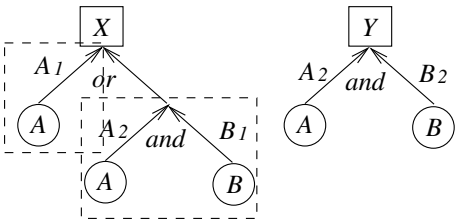

Fig. 1. Two example pattern trees

Conventional fuzzy rules can be extracted from pattern trees. For example, the following rules can be obtained from the pattern trees as in Fig. 1.

$$Rule1 : IF A = A_1 \; THEN \; class = X \quad (5)$$
$$Rule2 : IF A = A_2 \; AND \; B = B_1 \; THEN \; class = X \quad (6)$$
$$Rule3 : IF A = A_2 \; AND \; B = B_2 \; THEN \; class = Y \quad (7)$$

In addition to the conventionally used fuzzy aggregations MIN/MAX, pattern trees can choose any aggregations as described in subsection II-B.

III. Proposed Pattern Tree Induction Method

Without losing generality, assume a dataset has n input variables A_i, $i = 1, 2, \ldots, n$ and one output variable B. For representational simplicity, further assume both the input and output variables have m fuzzy linguistic terms denoted as A_{ij} and B_j, $i = 1, 2, \ldots, n$, and $j = 1, 2, \ldots, m$. That is, the

dataset is represented by a fuzzy membership value matrix of dimension $(n+1) \times m$. The task is to build m pattern trees for the m output classes (fuzzy terms). This section proposes a pattern tree induction method in pseudo code to build a pattern tree for class B_1. The pattern trees for other classes can be built following the same procedure.

Algorithm 1 shows the class *TreeNode* which represents the nodes such as the dotted boxes as shown in Fig. 1. *FuzzyTerms[2]* stores either one (for non-bottom nodes) or

Algorithm 1 class TreeNode{

```
1:  FuzzyTerms[2] usedFuzzyTerms
2:  int aggreOpe
3:  double[] aggreVals
4:  double similarity
5:  TreeNode child
6:  TreeNode parent
7: }
```

two (for bottom node) fuzzy terms (such as A_{12}) which are used to aggregate. Note that $FuzzyTerm$ is represented by two integers which are the indices of the fuzzy term. For instance, $\{1, 2\}$ stands for fuzzy term A_{12}. *aggreOpe* indicates which aggregation operator is used, *aggreVals* stores the aggregated values on this node, *similarity* is the similarity measure between *aggreVals* and the output class (B_1), *child* points to this node's child node, and *parent* to its parent node.

Algorithm 2 is the main algorithm which builds a leaf node and then invokes the *buildPatternTree()* method to build the rest of the pattern tree. In particular, a similarity matrix S is calculated at line 1 by method *calSimilarity()*. S is of dimension $n \times m$, with each element being the similarity measure of the input fuzzy terms A_{ij}, $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, m$, and the output class B_1, calculated by (2) or (4). $simMax$ is the max value in the similarity matrix S, and $iMax$ and $jMax$ are the indices of the input variables and terms which together point to the max value $simMax$. The leaf node $leaf$ is created from line 4 and the indices of fuzzy term $\{iMax, jMax\}$ are assigned to *leaf*'s used fuzzy term $usedFuzzyTerm[0]$. The fuzzy term values $A_{iMax,jMax}$ and similarity value $simMax$ are copied to the leaf node. The *null* value is assigned to leaf's child as it does not have a child node. Then the similarity measure by indices of $\{iMax, jMax\}$ in the similarity matrix S is marked -1 to prevent this fuzzy term to be used in further construction of the pattern tree. Finally, $buildPatternTree()$ is invoked to build the whole tree and then the root node $root$ is returned.

The $buildPatternTree()$ method is shown in algorithm 3. It takes a leaf node and a similarity matrix as inputs and outputs the root node of the constructed pattern tree. It recursively invokes itself when building the pattern tree.

When a node (either the bottom leaf or not) is passed into this method, the node's similarity and aggregation values are assigned to variables *maxSim* and *currentAggreVals*. The boolean variable *selected* is initialized with *false* indicating that no fuzzy term so far has been found to aggregate with this node's aggregation values *currentAggreVals* (fuzzy

Algorithm 2 main()

Input: A fuzzified dataset ds
Output: The root of the constructed pattern tree for class B_1

```
1:  S = calSimilarity(ds)
2:  simMax = max{S}
3:  {iMax, jMax}=argmax{S}
4:  TreeNode leaf = new TreeNode()
5:  leaf.usedFuzzyTerms[0] ={iMax, jMax}
6:  leaf.aggreVals = A_{iMax,jMax}
7:  leaf.similarity = simMax
8:  leaf.child = null
9:  S(iMax, jMax) = -1
10: root = buildPatternTree(leaf, S)
```

Algorithm 3 buildPatternTree()

Input: A tree node $node$ and similarity matrix S
Output: the root of the constructed pattern tree

```
1:  maxSim = node.similarity
2:  currentAggreVals = node.aggreVals
3:  boolean selected = false
4:  for i = 1 to n do
5:    for j = 1 to m do
6:      //fuzzy term not used before
7:      if S(i, j)! = −1 then
8:        //try to associate fuzzy term A_ij
9:        //go through all available aggregation operators
10:       for k = 1 to numAggreOpes do
11:         tempAggreVals = aggregate(k,
                                currentAggreVals, A_ij, B_1)
12:         tempSim = similarity(tempAggreVals, B_1)
13:         if maxSim < tempSim then
14:           maxSim = tempSim
15:           {iMax, jMax} = {i, j}
16:           maxSimAggreVals = tempAggreVals
17:           aggregationOpe = k
18:           selected = ture
19:         end if
20:       end for
21:     end if
22:   end for
23: end for
24: if selected == true then
25:   //leaf node
26:   if node.child == null then
27:     node.usedFuzzyTerm[1] = {iMax, jMax}
28:   else
29:     node.usedFuzzyTerm[0] = {iMax, jMax}
30:   end if
31:   node.aggreOpe = aggregationOpe
32:   node.aggreVals = maxSimAggreVals
33:   node.similarity = maxSim
34:   node.parent = new TreeNode()
35:   node.parent.child = node
36:   S(iMax, jMax) = −1
37:   return buildPatternTree(node.parent, S)
38: else
39:   return node.child
40: end if
```

values of the fuzzy term chosen in the *main()* method for the first invoking time).

For all similarity measures in the similarity matrix S (two **for** loops at lines 4 and 5), if the fuzzy term has not been used before in building pattern tree (line 7), then it is considered as a candidate to aggregate with this node, via different available aggregation operators (line 10) as described in subsection II-B. For each aggregation operator, the aggregation values *tempAggreVals* are calculated by *aggregate()* method (see Algorithm 4), which takes four arguments as inputs, namely, the index of aggregation operator, the current aggregation

values, the candidate fuzzy term values, and the class term values (B_1). The similarity measure (*tempSim* at line 12) can thus be calculated from the candidate aggregated values and the output class (B_1) values, via the similarity calculation as shown in (2) or (4). Among the combinations of all candidate fuzzy terms and aggregation operators, the one which results in the max similarity measure is chosen. The information about the combination are assigned to *maxSim, iMax, jMax, maxSimAggreVals*, and *aggregationOpe* respectively. Since a combination has been chosen, the boolean variable *selected* is set *true*.

If one combination of fuzzy term and aggregation operator is chosen after the loop ending at line 23, all the combination information is copied to the node from line 26 to 33. It is worth noting that if the node is a leaf node (line 26), then the selected fuzzy term is copied to *node*.usedFuzzyTerm[1] rather than *node*.usedFuzzyTerm[0]. This is because the *node*.usedFuzzyTerm[0] of the bottom leaf node has already stored the fuzzy term which was chosen in the *main()* method (algorithm 2). The method then keeps invoking itself (line 37) to aggregate at its parent node until no combination is chosen for aggregation, in that case (line 39), the node's child is returned as the root of the constructed pattern tree.

The method *aggregate()* is given in algorithm 4. It shows how to calculate the aggregated values when the aggregation operator and two vectors of fuzzy term values are given. This method additionally requires the use of the class fuzzy values if the aggregation operator is OWA or WA.

Algorithm 4 aggregate()

Input: Aggregation operator $aggOpe$, two vectors of fuzzy term values $aggreVals1$, $aggreVals2$, and output class values $classVals$ (required if the operator is WA or OWA)
Output: Aggregated values $aggregatedVals$

```
1:  if aggOpe!= OWA or aggOpe!= OWA then
2:     for i = 1 to sizeOf(aggreVals1) do
3:        aggregatedVals[i] =
                        aggOpe(aggreVals1[i], aggreVals2[i])
4:     end for
5:  else if aggOpe == OWA then
6:     for i = 1 to sizeOf(aggreVals1) do
7:        maxVals[i] = max{aggreVals1[i], aggreVals2[i]}
8:        minVals[i] = min{aggreVals1[i], aggreVals2[i]}
9:     end for
10:    λ = calLambda(maxVals, minVals, classVals)
11:    for i = 1 to sizeOf(aggreVals1) do
12:       aggregatedVals[i] = λ * maxVals[i]+
                              (1 − λ) * minVals[i]
13:    end for
14: else if aggOpe == WA then
15:    λ = calLambda(aggreVals1, aggreVals2, classVals)
16:    for i = 1 to sizeOf(aggreVals1) do
17:       aggregatedVals[i] = λ * aggreVals1[i]+
                              (1 − λ) * aggreVals2[i]
18:    end for
19: end if
```

Method *calLambda()* calculates a singleton value λ at lines 10 and 15 so that the vector of $aggregatedVals[i]$ at line 12 and 17 has the closest distance from $classVals$ in the sense of the mean square error.

To make the pattern tree induction easier to understand, a small numerical example is given.

Example 2: Assume an artificial dataset has two input variables A and B, with each having two fuzzy linguistic terms A_i and B_i, $i = 1, 2$. Also assume this dataset has two output classes X and Y. All the fuzzy membership values are shown in Table II. To simplify the representation, only

TABLE II
AN 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

the construction of pattern tree for class X is presented and the MIN/MAX, OWA, and WA aggregations are considered here. The construction includes the following steps:

1) Calculate the similarity matrix which contains the similarity measures between fuzzy terms A_1, A_2, B_1, B_2 and X by using (2). They are 0.6585, 0.4348, 0.4545, and 0.5217 respectively.
2) Assign fuzzy term A_1 which has the maximal similarity measure to the bottom leaf node.
3) Try the rest of the fuzzy terms A_2, B_1, and B_2 in turn to aggregate with A_1 using different aggregation operators MIN, MAX, OWA, and WA. The resulted similarity measures are shown in Table III

TABLE III
SIMILARITY MEASURES OF CANDIDATE COMBINATIONS IN STEP 3)

	A_2	B_1	B_2
MIN	0.3500	0.3000	0.5610
MAX	0.7021	0.7778	0.6087
OWA	0.6815	0.7740	0.6065
WA	0.6356	0.6543	0.6585

4) Choose fuzzy term B_1 and MAX operator to aggregate fuzzy term A_1. This is because the aggregation of B_1 with MAX operator results in the maximal similarity of 0.7778 which is greater than the similarity produced by A_1 alone. Also, the aggregated values of $MAX(A_1, B_1) = \{0.8, 0.9, 0.9, 0.8, 0.4, 0.5\}$ are stored in the leaf node for further aggregation.
5) Try the rest of the fuzzy terms A_2 and B_2 in turn to aggregate with $MAX(A_1, B_1)$ using different aggregation operators. The resulting similarity measures are shown in Table IV.

TABLE IV
SIMILARITY MEASURES OF CANDIDATE COMBINATIONS IN STEP 5)

	A_2	B_2
MIN	0.4762	0.5349
MAX	0.7143	0.7500
OWA	0.7109	0.7609
WA	0.7778	0.7792

6) Choose the fuzzy term B_2 and WA operator to aggregate the fuzzy term $MAX(A_1, B_1)$, as the aggregation of B_2 with WA operator results in the maximal similarity 0.7792 which is greater than the similarity produced by $MAX(A_1, B_1)$. Note that a $\lambda = 0.9074$ is stored in the node along with the WA operator. Also, the aggregated values of $WA(MAX(A_1, B_1), B_2, \lambda) =$

$\{0.8185, 0.9000, 0.8259, 0.7444, 0.41856, 0.5\}$ are stored in this node for further aggregation.

7) Try the rest of the fuzzy term A_2 to aggregate with $WA(MAX(A_1, B_1), B_2, \lambda)$ using different aggregation operators. None of the resulting similarity measures are greater than the current 0.7792. The pattern tree building process stops and the result is shown in Fig. 2.

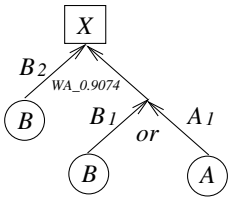

Fig. 2. Pattern tree generated for class X in example 2

IV. Comparison to SBM, WSBM and Decision Trees

As the fuzzy subsethood based method (SBM) [2] and its extension, the weighted subsethood based method (WSBM) [7], are two specific cases of patterns trees (see below), they are chosen along with the well-known fuzzy decision tree induction [13] to compare with the proposed pattern tree induction method. For a clear representation of the comparison, the small artificial dataset as shown in Table II is used to generate fuzzy rules for SBM, WSBM, decision trees induction, and pattern trees induction.

A. SBM

The subsethood based method consists of three main steps:

1) Classifying training data into subgroups according to class values, with each group having the data which prefer voting 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, $Sub(X, A_1) = \frac{X \cap A_1}{X}$, represents the degree to which X is a subset of A_1 (see [2], [7]), where $\cap$ is a t-norm operator (MIN is used).
3) Creating rules based on fuzzy subsethood values. In particular, for each input variable, the fuzzy term that has the maximal subsethood value (must greater than or equal to a pre-specified threshold value $\alpha \in [0, 1]$, 0.9 used in [2]) will be chosen as an antecedent for the resulting fuzzy rules.

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

For the artificial dataset, two groups are created, with the first having the first four data points, which prefer voting class X rather than Y, and the second having the remaining two data points. Assume 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 [2], no fuzzy rules (pattern trees) can be generated. However, for comparison purposes, α is set to 0.6 to generate the pattern tree for class X as shown in Fig. 3. This figure also shows the pattern tree for Y which is constructed in the same manner.

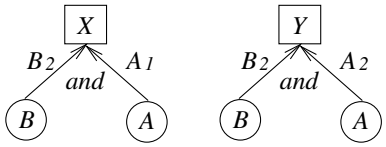

Fig. 3. Pattern trees generated by SBM

B. WSBM

WSBM is a weighted version of SBM. It uses a certain weighting strategy to represent fuzzy terms rather than choosing the one which has the greatest subsethood value per variable. In particular, the weight for fuzzy term A_i is defined as

$$w(X, A_i) = \frac{Sub(X, A_i)}{max_{j=1,2} Sub(X, A_j)}. \quad (8)$$

Using this equation, the weights for A_1, A_2, B_1 and B_2 in the classification of X are calculated as 1, 0.7, 0.8, and 1 respectively. The fuzzy rules generated by WSBM form fixed structured pattern trees as shown in Fig. 4.

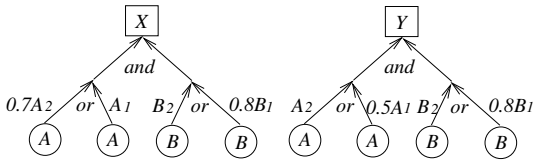

Fig. 4. Pattern trees generated by WSBM

Unlike the proposed pattern tree induction method which can generate different structured patter trees, SBM and WSBM generate trees with nearly fixed structures. In particular, SBM trees have only one level, on which the fuzzy terms which have the greatest subsethood values per input variable aggregate via a t-norm operator. WSBM trees have two levels. On the bottom level, all fuzzy terms (with different weights) for each variable aggregate via a t-conorm operator, on the top level the aggregated values further aggregate via a t-norm operator. The fixed structures of SBM and WSBM may prevent them from obtaining high performance in classification applications (see subsection IV-E).

C. Fuzzy Decision Tree Induction

The decision tree induction [5] is a classic machine learning method. It has been extended to induce fuzzy decision tree by Yuan and Shaw [13], where the fuzzy entropy is used to guide the search of most effective branches. A fuzzy decision tree as shown in Fig. 5 can be built using [13] over the artificial dataset. It has a root on the top and several leaf nodes on the bottom. When a new data sample needs to be classified by the fuzzy decision tree, it starts from the root

and travels down 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.

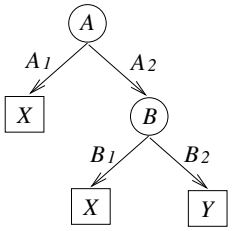

Fig. 5. A decision tree generated by fuzzy decision tree induction

For the same training dataset, fuzzy decision tree induction may generate different results with different numbers of leaf nodes, which are used as criteria to terminate the tree building process. Considering only six data samples are available, this can be set to 1 to 6. Among all these, the best result (as shown in Fig. 5, when the number of leaf nodes is set to 1 or 2) has been chosen for comparison in subsection IV-E.

D. Relation between Fuzzy Decision Trees and Pattern Trees

A fuzzy decision tree can be converted to a set of pattern trees whose size is equal to the number of output classes. In particular, the easiest way is to convert a fuzzy decision tree to a set of semi-binary pattern trees. That is, there are maximally two branches allowed for each node (except for the root) in the pattern tree. The conversion process 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 levels of the pattern tree. A fuzzy rule consisting of n fuzzy terms results in a $(n-1)$-level binary pattern tree as the bottom level contains two fuzzy terms. The fuzzy rules which have the same classification class are connected by a t-conorm (usually MAX) at the top level (level n) to construct a pattern tree for this output class. The number of levels of the generated semi-binary pattern trees remains the same as the fuzzy decision tree, regardless whether the decision tree is binary or not. Fig. 1 shows the pattern trees converted from the decision tree as shown in Fig. 5. They both have the same rule base as listed in (5) - (7).

The conversion from a decision tree to pattern trees is not unique. For example, an alternative conversion from Fig. 5 is shown in Fig. 6, which can be represented by two rules as following.

$$Rule1: IF A = A_1 \ OR \ B = B_1 \ THEN \ class = X, \quad (9)$$

$$Rule2: IF A = A_2 \ AND \ B = B_2 \ THEN \ class = Y. \quad (10)$$

Note that these two fuzzy rules are functionally equal to rules (5), (6) and (7). When the size of fuzzy terms for each variable increases, the conversion of a fuzzy decision tree to multi-branch pattern trees are desirable and the work on that is on-going. However, this paper mainly focuses on the induction and use of binary pattern trees.

On the other hand, pattern trees can be converted to a decision tree although the conversion does not seem so straightforward. The fuzzy decision tree shown in Fig. 5 can be converted from either Fig. 1 or Fig. 6.

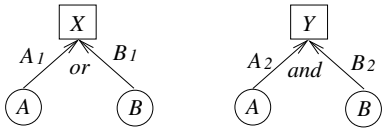

Fig. 6. Alternative pattern trees converted from the decision tree in Fig. 5

It is worth noting that pattern trees are not just inverted decision trees; They are different as: 1) the former focus on separating data samples which have *different* output classes, whist the latter focus on representing the structures of data samples which have the *same* output classes; 2) the former consider *all* fuzzy terms of the chosen input variable in building a tree, whist the latter only consider *one* fuzzy term of the chosen input variable; and 3) the former normally only make use of the MIN and MAX aggregations, while the latter can use any aggregations as described in subsection II-B.

E. Pattern Trees and the Comparison Results

The MIN/MAX, algebraic AND/OR, Łukasiewicz, EINSTEIN, OWA, and WA are considered in building a pattern tree using the proposed method. Two similarity measures as shown in (2) and (4) are used and the best result (using 4) is shown in Fig. 7. In the case that if only MIN/MAX

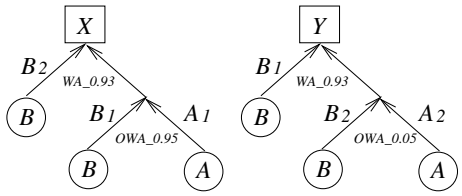

Fig. 7. Pattern trees generated by proposed method

are allowed in the pattern trees, the generated trees using similarity defined in either (2) or (4) are exactly the same as the trees shown in Fig. 6, with and being MIN and or being MAX.

Now apply this artificial dataset to the rules (or trees) generated by SBM, WSBM, fuzzy decision tree and pattern trees (with A indicating only MIN/MAX are considered and B all aggregations are considered), the results including the number of correctly predicted data samples (correct No), the root mean square error (RMSE) of the predictions for class X, Y, and their average (ARMSE) are shown in Table V.

TABLE V
RESULTS OF SBM, WSBM, FUZZY DECISION TREE, AND PATTER TREES USING AN ARTIFICIAL DATASET

	Correct No	RMSE(X)	RMSE(Y)	ARMSE
SBM	4	0.3916	0.2	0.2958
WSBM	4	0.2386	0.3435	0.2911
Fuzzy decision tree	5	0.2	0.2	0.2
Pattern trees (A)	5	0.2	0.2	0.2
Pattern trees (B)	6	0.1961	0.1956	0.1959

It is clear that the pattern trees (B) perform the best, in terms of both the correctly predicted number and the mean

square error, among all the methods. Pattern trees (A) performs slightly worse (but as good as the fuzzy decision tree) as it only uses MIN/MAX aggregations, revealing that other aggregations can be more suitable than MIN/MAX in some cases. This example shows that SBM and WSBM cannot get good results as their fixed pattern tree structures prevent them appropriately representing the structure of the dataset. Fuzzy decision trees can generate different tree structures. However, they lack some candidate search spaces represented by the t-conorm aggregations of fuzzy terms of different variables. For instance, in this example, Fig. 5 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 pattern tree induction method explicitly considers both t-norm and t-conorm aggregations of fuzzy terms in building trees. Thus, it is more likely to find optimal solutions.

V. Experimental Results

To demonstrate the effectiveness of the proposed pattern tree induction method, comparative studies with the fuzzy decision tree [13] have been carried out via experiments using the datasets obtained from UCI machine learning repository [4], which have been widely used as benchmarks in classification applications. The datasets include Iris-Plant, Wisconsin Breast Cancer, Diabetes and Wine Recognition as summarized in Table VI. Note that the results of SBM and WSBM over the Iris-Plant dataset, which were reported in [7], are also provided here for comparison.

TABLE VI
Experimental datasets summary

Name	Number of data	Number of inputs	Number of classes
Iris-Plant	150	4	3
Wisconsin Breast Cancer	699	9	2
Diabetes	768	8	2
Wine Recognition	178	13	3

For all datasets except Iris-Plant, 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. To be comparable to the result reported in [7], Iris-Plant dataset uses three evenly distributed trapezoidal fuzzy sets for each variable.

All datasets (*ds*) are divided into training datasets and test ones. Assume every dataset is labeled for data samples, the training sets (*ds-odd*) contain the odd numbered data samples and the test sets (*ds-even*) contain the even numbered ones. The performances of the SBM and WSBM (only for Iris-Plant dataset), fuzzy decision tree and pattern trees over different combinations of training-test sets for different datasets are shown in Table VII, VIII, IX and X.

All the results reported for fuzzy decision trees are the best results among the use of different numbers of leaf nodes. The MIN/MAX, algebraic AND/OR, Łukasiewicz, EINSTEIN, OWA, and WA are considered in building pattern trees. Note that $*$ in Table VII indicates the results are not available in [7].

TABLE VII
Prediction accuracy of SBM, WSBM, Fuzzy decision tree, and patter trees using Iris-Plant dataset

Training dataset	Testing dataset	SBM accuracy	WSBM accuracy	Fuzzy decision tree accuracy	Pattern trees accuracy
ds-odd	ds-even	80%	93.33%	97.33%	97.33%
ds-even	ds-odd	78.67%	93.33%	97.33%	98.67%
ds	ds	*	*	97.33%	97.33%

TABLE VIII
Prediction accuracy of Fuzzy decision tree and pattern trees using Wisconsin Breast Cancer dataset

Training dataset	Testing dataset	Fuzzy decision tree accuracy	Pattern trees accuracy
ds-odd	ds-even	93.70%	94.84%
ds-even	ds-odd	95.71%	96.57%
ds	ds	96.85%	97.57%

TABLE IX
Prediction accuracy of Fuzzy decision tree and pattern trees using Diabetes dataset

Training dataset	Testing dataset	Fuzzy decision tree accuracy	Pattern trees accuracy
ds-odd	ds-even	75.26%	72.92%
ds-even	ds-odd	74.48%	72.14%
ds	ds	91.15%	76.04%

TABLE X
Prediction accuracy of Fuzzy decision tree and pattern trees using Wine Recognition dataset

Training dataset	Testing dataset	Fuzzy decision tree accuracy	Pattern trees accuracy
ds-odd	ds-even	92.13%	95.51%
ds-even	ds-odd	91.01%	97.75%
ds	ds	97.75%	98.31%

Out of the four datasets, the proposed pattern tree induction results in higher classification accuracy than fuzzy decision tree over datasets including Iris-Plant, Wisconsin Breast Cancer and Wine Recognition. Only in Diabetes dataset, fuzzy decision trees obtain better results than pattern trees.

More interestingly, it can be observed that pattern trees perform in a consistent way for different combinations of training and test datasets, while fuzzy decision tree does not. This is can be found from the Diabetes dataset. Fuzzy decision trees generate large difference in classification accuracy between the first (or the second) combination of the training-test datasets and the third one, due to the over-fitting problem. The reason is that decision tree induction considers only a portion of the whole training dataset in choosing the branches at the low levels of trees. The lack of using the whole training dataset inevitably prevents the method finding better tree structures for all the dataset. In contrast, pattern trees make use of the whole data in building each level of

the tree, which ensures the tree keeps good generality for classifications.

In addition, pattern trees usually have less complex structures than fuzzy decision trees. For example, Fig. 8 shows the decision tree generated using Wine Recognition dataset with training set being ds-odd and test set being ds-even. The ellipses are the input variables and the rectangles are

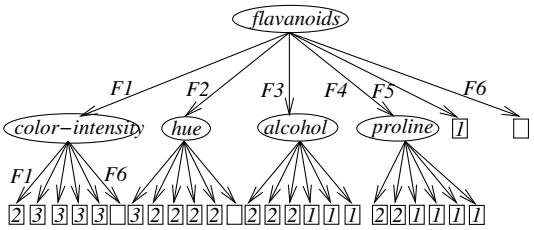

Fig. 8. Decision tree generated using Wine Recognition dataset

the output classes. Note that the empty rectangles mean no decision class is available. Fi, $i = 1, \ldots, 6$, are the fuzzy terms associated with each input variable. Fig. 9 shows one (for class 2) of the three pattern trees generated using the same training dataset. As there are two other classes in this

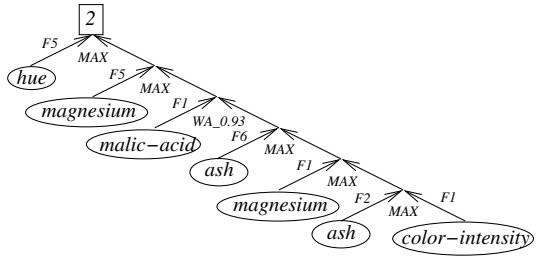

Fig. 9. Pattern tree generated for class 2 using Wine Recognition dataset

dataset, two other pattern trees (for class 1 and 3) are required for classifications. Even so, the pattern trees are normally less complex than the decision tree in terms of the size of leaf nodes. In this example, the pattern trees have $7 \times 3 = 21$ leaf nodes in total as each pattern tree per class has 7 fuzzy terms, while the decision tree has 26.

The complexity can also be compared in the form of rule representation. If only class 2 is considered, the rules extracted from the decision tree and from the pattern tree are listed in (11) and (12) respectively. It can be seen that the latter is simpler than the former.

$$\begin{aligned}
&(IF\ flavanoids = F_1\ AND\ \textit{color-intensity} = F_1)\ OR\\
&\quad (IF\ flavanoids = F_2\ AND\ hue = F_2)\ OR\\
&\quad (IF\ flavanoids = F_2\ AND\ hue = F_3)\ OR\\
&\quad (IF\ flavanoids = F_2\ AND\ hue = F_4)\ OR\\
&\quad (IF\ flavanoids = F_2\ AND\ hue = F_5)\ OR\\
&\quad (IF\ flavanoids = F_3\ AND\ alcohol = F_1)\ OR\\
&\quad (IF\ flavanoids = F_3\ AND\ alcohol = F_2)\ OR\\
&\quad (IF\ flavanoids = F_3\ AND\ alcohol = F_3)\ OR\\
&\quad (IF\ flavanoids = F_4\ AND\ proline = F_1)\ OR\\
&\quad (IF\ flavanoids = F_4\ AND\ proline = F_2)\\
&\quad THEN\ class = 2 \qquad (11)\\
&((((((IF\ \textit{color-intensity} = F_1\ MAX\ ash = F_2)\ MAX\\
&\quad magnesium = F_1)\ MAX\ ash = F_6)\ WA\\
&\quad \textit{malic-acid} = F_1)\ MAX\ magnesium = F_5)\ MAX\\
&\quad hue = F_5)\\
&\quad THEN\ class = 2 \qquad (12)
\end{aligned}$$

VI. Conclusions

This paper has proposed a new type of tree termed pattern trees which make use of different aggregations (optional). Like decision trees, pattern trees are an effective tool for classification applications. This paper has discussed the difference between decision trees and pattern trees. It has also shown that SBM and WSBM are two specific cases of pattern trees, with each having a fixed pattern tree structure.

A novel pattern tree induction method has been proposed to build the pattern trees. The comparison to other classification methods including SBM, WSBM and fuzzy decision tree shows that pattern trees can obtain higher accuracy rates in classifications. It also shows that pattern trees perform more consistently than fuzzy decision trees. They are capable of generating patterns with good generality, while fuzzy decision trees can easily fall into the trap of over-fitting.

Although the proposed pattern tree induction method shows very promising results, it still needs improvement. In particular, the proposed method only generates binary pattern trees, the enhancement to generate multi-branch pattern trees is highly desirable. In addition, the conversion between decision trees and pattern trees is worth investigating.

Acknowledgment

The authors would like to thank Professor László Kóczy, Sumudu U. Mendis and János Botzheim for their helpful discussions on the work presented here.

References

[1] Chao, C. T., Chen, Y. J., and Teng, C. C., "Simplification of fuzzy neural systems using similarity analysis," *IEEE Transactions on Systems, Man, and Cybernetics Part B*, vol. 26, no. 2, pp. 344 – 354, 1996.

[2] Chen, S. M., Lee, S. H. and Lee, C. H., "A new method for generating fuzzy rules from numerical data for handling classification problems," *Applied Artificial Intelligence*, vol. 15, pp. 645 – 664, 2001.

[3] Kóczy, L. T., Vámos, T. and Biró, G., "Fuzzy signatures," *EUROFUSE-SIC*, pp. 210 – 217, 1999.

[4] Newman, D. J., Hettich, S., Blake, C. L. and Merz, C.J., UCI Repository of machine learning databases [http://www.ics.uci.edu/ mlearn/MLRepository.html], 1998.

[5] Quinlan, J. R., "Decision trees and decision making," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 20, no. 2, pp. 339 – 346, 1994.

[6] Raju, G. V. S. and Kisner, R. A., "Hierarchical fuzzy control," *International Journal Control*, vol. 54, no. 5, pp. 1201 – 1216, 1991.

[7] Rasmani, K. A. and Shen, Q., "Weighted linguistic modelling based on fuzzy subsethood values," *IEEE International Conference on Fuzzy Systems*, vol. 1, pp. 714 – 719, 2003.

[8] Schweizer, B. and Sklar, A., "Associative functions and abstract semigroups," *Publ. Math. Debrecen*, vol. 10, pp. 69 – 81, 1963.

[9] Wang, L. X. and Mendel, J. M., "Generating fuzzy rules by learning from examples," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 6, pp. 1414 – 1427, 1992.

[10] Wang, L. X. and Mendel, J. M., "Fuzzy Basis functions, universal approximation, and orthogonal least-squares learning," *IEEE Transactions on Neural Networks*, vol. 3, no. 5, pp. 807 – 814, 1992.

[11] Wong, K. W, Gedeon, T. D. and Kóczy L. T., "Construction of fuzzy signature from data: an example of SARS pre-clinical diagnosis system," *IEEE International Conference on Fuzzy Systems*, vol. 3, pp. 1649 - 1654.

[12] Yager R. R., "On ordered weighted averaging aggregation operators in multicritera decison making," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 18, pp. 183 – 190, 1988.

[13] Yuan, Y. and M. J. Shaw, "Induction of fuzzy decision trees," *Fuzzy Sets and Systems*, vol. 69, no. 2, pp. 125 – 139, 1995.

[14] Zadeh, L. A., "Fuzzy sets," *Information and Control*, vol. 8, pp. 338 – 353, 1965.