Pattern Trees for Regression and Fuzzy Systems Modeling

Robin Senge and Eyke Hüllermeier
Department of Mathematics and Computer Sciences
Marburg University, Germany

Draft of a paper published in Proc. WCCI–2010, World Congress on Computational Intelligence, Barcelona, 2010.

Abstract

Fuzzy pattern tree induction has recently been introduced as a novel classification method in the context of machine learning. Roughly speaking, a pattern tree is a hierarchical, tree-like structure, whose inner nodes are marked with generalized (fuzzy) logical operators and whose leaf nodes are associated with fuzzy predicates on input attributes. In this paper, we adapt the method of pattern tree induction so as to make it applicable to another learning task, namely regression. Thus, instead of predicting one among a finite number of discrete class labels, we address the problem of predicting a real-valued target output. Apart from showing that fuzzy pattern trees are able to approximate real-valued functions in an accurate manner, we argue that such trees are also interesting from a modeling point of view. In fact, by describing a functional relationship between several input attributes and an output variable in an interpretable way, pattern trees constitute a viable alternative to classical fuzzy rule models. Compared to flat rule models, the hierarchical structure of patterns trees further allows for a more compact representation and for trading off accuracy against model simplicity in a seamless manner.

1 Introduction

Fuzzy pattern tree induction was recently introduced as a novel machine learning method for classification by Huang, Gedeon and Nikravesh [4]. Independently, the same type of model was proposed in [16] under the name "fuzzy operator tree". Roughly speaking, a fuzzy pattern tree is a hierarchical, tree-like structure, whose inner nodes are marked with generalized (fuzzy) logical and arithmetic operators, and whose leaf nodes are associated with fuzzy predicates on input attributes. A pattern tree propagates information from the bottom to the top: A node takes the values of its descendants as input, combines them using the respective operator, and submits the output to its predecessor. Thus, a pattern tree implements a recursive mapping producing outputs in the unit interval.

In this paper, we adapt the method of pattern tree induction so as to make it applicable to another learning task, namely regression. Thus, instead of predicting one among a finite number of discrete class labels, the problem is to predict a real-valued target output. Apart from showing that fuzzy pattern trees are able to approximate real-valued functions in an accurate manner, we argue that such trees are also interesting from a modeling point of view. In fact, by describing a functional relationship between several input attributes and an output variable in an interpretable way, pattern trees constitute a viable alternative to classical fuzzy rule models. Compared to flat rule models, the hierarchical structure of patterns trees further allows for a more compact representation and for trading off accuracy against model simplicity in a seamless manner.

The remainder of the paper is structured as follows. Section II gives a brief introduction to the model class of pattern trees and shows how it can be used for representing regression functions. The learning of such trees in a data-driven way is addressed in Section III, where an algorithm for constructing pattern trees in a top-down manner is proposed. In Section IV, we explain how pattern trees can be used in a more general way for fuzzy systems modeling. Section 5 is devoted to an experimental study evaluating our method in terms of predictive accuracy. The paper ends with some concluding remarks in Section 6.

2 Pattern Tree Models

As will be explained in more detail in this section, a pattern tree implements a mapping from n input variables to one output variable:

$$\begin{aligned} f : \mathbb{X}_1 \times \mathbb{X}_2 \times \ldots \times \mathbb{X}_n &\rightarrow [0,1] \\ (x_1, x_2, \ldots, x_n) &\mapsto y \end{aligned}$$

Here, $\mathbb{X}_i$ is the domain of the i-th input variable X_i $(1 \leq i \leq n)$.

Note that the range of the function is the unit interval, i.e., the output variable assumes values in $[0, 1]$. One can think of the output value as the degree of membership of a fuzzy subset G of an underlying domain $\mathbb{Y}$. For example, if $\mathbb{Y}$ is an interval $[a, b]$, i.e., if the original output variable is lower-bounded by a and upper-bounded by b, then the membership function could be given by a simple linear scaling

$$G : y \mapsto \frac{y - a}{b - a} \quad . \tag{1}$$

Thus, the corresponding fuzzy set could be interpreted as a model of the linguistic term "large". Likewise, if the original output is unbounded, a possible re-scaling is

$$G : y \mapsto \frac{1}{1 + \exp(-\alpha y)} \quad . \tag{2}$$

More generally, G can be any fuzzy subset of $\mathbb{Y}$. In principle, the domain $\mathbb{Y}$ could also be partitioned into several fuzzy subsets, like "small", "medium" and "large". In this case, a single pattern tree will be needed for each of these fuzzy sets, and a defuzzification step is needed to produce the final output; we shall elaborate on this idea in more detail in Section V.

Considering the fuzzy set G as a fuzzy predicate or, say, property of the output variable (e.g., being large), a fuzzy pattern tree can be seen as a model that specifies criteria on the input attributes which imply this property to hold. From a modeling point of view, the pattern tree approach is based on three important conceptions:

- fuzzification of input attributes;
- hierarchical structuring of a functional dependency through recursive partitioning of criteria into sub-criteria;
- flexible aggregation of sub-criteria by means of parameterized fuzzy operators.

2.1 Fuzzification of Input Attributes

On the lowest level, the original input attributes (e.g., measurements) are modulated in terms of associated fuzzy sets. Consequently, the actual input of the pattern tree is not a feature itself, but the degree to which is satisfies a certain fuzzy property. More specifically, each domain $\mathbb{X}_i$ is discretized by means of a fuzzy partition, that is, a set of fuzzy subsets

$$F_{i,j} : \mathbb{X}_i \rightarrow [0,1] \qquad (j = 1, \ldots, n_i)$$

such that $\sum_{j=1}^{n} F_{i,j}(x) > 0$ for all $x \in \mathbb{X}_i$. The $F_{i,j}$ are often associated with linguistic labels such as “small” or “large”, in which case they are also referred to as *fuzzy terms*.

2.2 Hierarchical Structure of the Model

A pattern tree is a hierarchical, tree-like structure in which information is processed from the bottom (leaf nodes) to the top (root node).[1] The input of a pattern tree is entered at the leaf nodes. More specifically, a leaf node is labeled by an attribute A_i and a fuzzy subset $F_{i,j}$ of the corresponding domain $\mathbb{X}_i$. Given an instance $\boldsymbol{x} = (x_1, \ldots, x_m) \in \mathbb{X}$ as an input, the node produces $F_{i,j}(x_i)$ as an output, that is, the degree of membership of x_i in $F_{i,j}$.

The results of the evaluations of internal nodes are propagated to the parents of these nodes in a recursive way. The output eventually produced by a pattern tree is given by the output of its root node. Fig. 1 shows some exemplary pattern trees.

Recalling the interpretation of the output produced by a pattern tree as a fuzzy degree to which a specific property is satisfied, it can be seen that the model simplifies the overall evaluation of this property by evaluating different sub-criteria first and aggregating these evaluations afterward. In other words, each subtree of a pattern tree can be seen as a criterion or, say, a high-level feature constructed from a number of low-level features (namely the original input attributes), and these criteria are combined by means of fuzzy logical operators in a recursive way. This kind of hierarchical modeling is intuitively appealing and commonly used in many fields, for example in decision making [10].

[1]This distinguishes them from classification and regression trees [9, 3, 6], which process information in the reverse direction: they assume an input at the root node and output a prediction at each leaf.

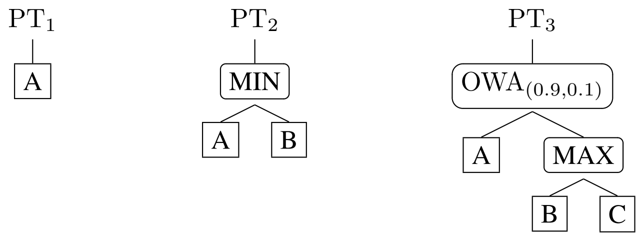

Figure 1: Examples of simple pattern trees.

Table 1: Fuzzy operators: T-norms

name	definition	code
Minimum	$\min\{a, b\}$	MIN
Algebraic	ab	ALG
Lukasiewicz	$\max\{a + b - 1, 0\}$	LUK
Einstein	$\frac{ab}{2-(a+b-ab)}$	EIN

2.3 Logic and Arithmetic Operators

As mentioned above, internal nodes of a pattern tree are labeled by generalized logical or arithmetic operators, including

- t-norms and t-conorms [7],
- weighted and ordered weighted average [11, 15].

The t-norms and t-conorms allowed as operators in our implementation of pattern tree induction are shown in Table 1 and Table 2, respectively.

Recall that an ordered weighted average (OWA) combination of k numbers $v_1, v_2, \ldots, v_k$ is defined by

$$\text{OWA}_w(v_1, v_2, \ldots, v_k) \overset{\text{df}}{=} \sum_{i=1}^{k} w_i \cdot v_{\tau(i)}, \tag{3}$$

where τ is a permutation of $\{1, 2, \ldots, k\}$ such that $v_{\tau(1)} \leq_{\tau(2)} \leq \cdots \leq v_{\tau(k)}$ and $w = (w_1, w_2, \ldots, w_k)$ is a weight vector satisfying $w_i \geq 0$ for $i = 1, 2, \ldots, k$ and $\sum_{i=1}^{k} w_i = 1$. Thus, just like the normal weighted average (WA), an OWA operator is parameterized by a set of weights. However, a weight does not directly refer to an

Table 2: Fuzzy operators: T-conorms

name	definition	code
Maximum	$\max\{a, b\}$	MAX
Algebraic	$a + b - ab$	COALG
Lukasiewicz	$\min\{a + b, 1\}$	COLUK
Einstein	$\frac{a+b}{1+ab}$	COEIN

attribute, like in WA, but instead to a rank: w_i is the weight of the i-th smallest value among $v_1, v_2, \ldots, v_k$.

Note that for $k = 2$, (3) is simply a convex combination of the minimum and the maximum. In fact, the minimum and the maximum operator are obtained, respectively, as the two extreme cases of (3): $w_1 = 1$ yields

$$\mathrm{OWA}_w(v_1, \ldots, v_k) = v_{\tau(1)} = \min(v_1, \ldots, v_k)$$

and $w_k = 1$ gives

$$\mathrm{OWA}_w(v_1, \ldots, v_k) = v_{\tau(k)} = \max(v_1, \ldots, v_k).$$

Therefore, the class of OWA operators nicely "fills the gap" between the largest conjunctive combination, namely the minimum t-norm, and the smallest disjunctive combination, namely the maximum t-conorm.

3 Learning Pattern Trees for Regression

A method for learning pattern tree models in a data-driven way was originally introduced in [4]. Moreover, an alternative to this method was recently proposed in [12]. This alternative modifies the original algorithm in several ways. Notably, pattern trees are induced in a top-down instead of a bottom-up manner. As shown by the authors, this strategy leads to improved performance.

The aforementioned algorithms both address the problem of classification. In this section, we propose a variant of the top-down method proposed in [12] which is suitable for solving regression problems, i.e., problems with a real-valued target variable.

3.1 Basic Algorithm

Our top-down algorithm for learning pattern trees for regression, PT-reg, is presented in pseudo-code in Fig. 2. Roughly speaking, it implements a beam search in the space of pattern trees, maintaining the B best models so far ($B = 5$ is used as a default value). The basic steps of the approach are as follows:

1. initialize with primitive pattern trees
2. filter candidates by evaluation of their performance on the training data
3. check stopping criterion
4. generate new candidates through local search
5. loop at step 2

The algorithm starts by computing the set of all primitive pattern trees $\mathbf{P}$, namely pattern trees consisting of only a single root node, labeled by a fuzzy set $F_{i,j}$. Additionally, the first candidate set, $\mathbf{C}^0$, is initialized by the B best basic pattern trees, i.e., the trees with highest evaluation (see Section 3.2 below).

Pattern Tree Algorithm for Regression

```
1: {Initialization}
2: $\mathbf{P} = \{A_{ij}\}, i = 1, ..., n; j = 1, ..., m$
3: $\mathbf{C}^0 = argminB_{P \in \mathbf{P}}[L(P)]$
4: $\epsilon = 0.0025$
5: $t = 0$
6: {Induction}
7: {Loop on iterations}
8: $stop = false$
9: while not stop do
10:   $t = t + 1$
11:   $\mathbf{C}^t = \mathbf{C}^{t-1}$
12:   {Loop on each candidate}
13:   for all $C_i^{t-1} \in \mathbf{C}^{k-1}$ do
14:     {Loop on each leaf of the chosen candidate}
15:     for all $l_{chosen} \in leafs(C_i^{t-1})$ do
16:       {Loop on each available operator $\psi$}
17:       for all $\psi \in \Psi$ do
18:         {Loop on nearly each primitive pattern tree}
19:         for all $P \in \mathbf{P} \backslash l_{chosen}$ do
20:           $\mathbf{C}^t = \mathbf{C}^t \cup ReplaceLeaf(C_i^{t-1}, l_{chosen}, \psi, P)$
21:         end for
22:       end for
23:     end for
24:   end for
25:   $\mathbf{C}^t = argminB_{C_i^t \in \mathbf{C}^t}[L(C_i^t)]$
26:   if $min_{C_i^t \in \mathbf{C}^t}(L(C_i^t)) < (1 + \epsilon) min_{C_i^{t-1} \in \mathbf{C}^{t-1}}(L(C_i^{t-1}))$ then
27:     $stop = true$
28:   end if
29: end while
30: return $argmin_{C_i^t \in \mathbf{C}^t}[L(C_i^t)]$
```

Figure 2: Pattern tree algorithm for regression.

After initialization, the algorithm iterates over all candidate trees. Starting from line 11, it tries to improve the currently selected candidate C_i^{t-1} in terms of performance. To this end, new candidates are created by tentatively replacing exactly one leaf node L (labeled by a fuzzy term) of C_i^{t-1} by a new subtree. This new subtree is a basic pattern tree, namely a tree that combines two fuzzy terms of two attributes (one of which is given by L) by means of an operator (see Fig. 3 for an illustration). The new candidate tree thus obtained is then evaluated by computing its performance on the training data. Having tried all possible replacements of all leaf nodes of the trees in $\mathbf{C}^i$, the B best

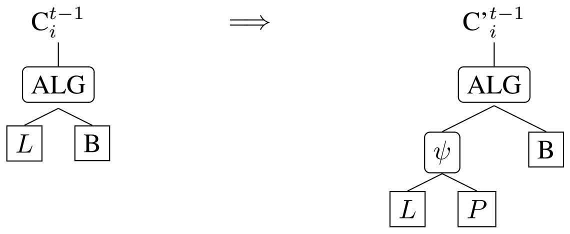

Figure 3: Top-down induction: A leaf node is expanded through replacement by a basic tree.

candidates are selected and passed to the next iteration, unless the termination criterion is fulfilled.

3.2 Performance Evaluation

To evaluate the performance of a pattern tree, we compute the squared error loss that is produces on the training data

$$\mathcal{T} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{n} \subset \mathbb{X} \times [0,1] \ .$$

Thus, with $f(\cdot)$ denoting the function implemented by the tree, we derive

$$L(f) = \frac{1}{n} \sum_{i=1}^{n} \left(f\left(\boldsymbol{x}^{(i)}\right) - y^{(i)} \right)^2 \ . \tag{4}$$

Note that a well-known disadvantage of the squared error loss, namely its sensitivity toward outliers, is less problematic in our case. In fact, since the output values are bounded by 0 and 1, the same holds true for the squared loss. In combination with a transformation like (2), which is needed to handle output variables with unbounded range, our approach can thus be seen as a kind of robust regression technique. Indeed, the combination of (2) and (4) produces an effect quite comparable to Huber's loss function, which combines the absolute (L_1) error for large differences with the squared error (L_2) for small ones [5].

3.3 Termination Criterion

The termination decision is based on the *relative* improvement of the best model in the $(t+1)$-st iteration (i.e., the model with the lowest loss (4)) as compared to the t-th iteration. More specifically, our algorithm stops if

$$L_{min}^{t+1} > (1-\epsilon) L_{min}^{t} \ , \tag{5}$$

i.e., if the relative improvement is smaller than ϵ, where $\epsilon \in (0,1)$ is a user-defined parameter. Based on empirical evidence, we propose $\epsilon = 0.001$ as a suitable value for this parameter.

3.4 Fuzzy Partitions

To make pattern tree learning amenable to numeric attributes, these attributes have to be "fuzzified" and discretized beforehand. Fuzzification is needed because fuzzy logical operators at the inner nodes of the tree expect values between 0 and 1 as input, while discretization is needed to limit the number of candidate trees in each iteration of the learning algorithm. Besides, fuzzification may also support the interpretability of the model.

Fuzzy partitions can of course be defined in various ways. In our implementation, we discretize a domain $\mathbb{X}_i$ in a generic way, using three fuzzy sets $F_{i,1}, F_{i,2}, F_{i,3}$ associated, respectively, with the terms "low", "medium" and "high". The first and the third fuzzy set are defined as

$$F_{i,1}(x) = \begin{cases} 1 & x < min \\ 0 & x > max \\ 1 - \frac{x-min}{max-min} & otherwise \end{cases},$$

$$F_{i,3}(x) = \begin{cases} 1 & x > max \\ 0 & x < min \\ \frac{x-min}{max-min} & otherwise \end{cases},$$

with min and max being the minimum and the maximum value of the attribute in the training data. Noting that all operators appearing at inner nodes of a pattern tree are monotone increasing in their arguments, it is clear that these fuzzy sets can capture two types of influence of an attribute on the output variable, namely a positive and a negative one.

The fuzzy set $F_{i,2}$ is meant to capture non-monotone dependencies. It is defined as a triangular fuzzy set with center c:

$$F_{i,2}(x) = \begin{cases} 0 & x \leq min \\ \frac{x-min}{c-min} & min < x \leq c \\ 1 - \frac{x-c}{max-c} & c < x < max \\ 0 & x \geq max \end{cases} \tag{6}$$

The parameter c is determined so as to maximize the absolute (Pearson) correlation between the membership degrees of the attribute values in $F_{i,2}$ and the corresponding output variable on the training data. In case the correlation is negative, $F_{i,2}$ is replaced by its negation $1 - F_{i,2}$.

Finally, nominal attributes are modeled as degenerate fuzzy sets: For each value v of the attribute, a fuzzy set with membership function

$$Term_v(x) = \begin{cases} 1 & x = v \\ 0 & otherwise \end{cases}$$

is introduced.

4 Fuzzy Systems Modeling

As already mentioned, fuzzy pattern trees can be considered as an interesting approach to fuzzy systems modeling and, in this regard, as an alternative to conventional fuzzy rule models. Although a detailed treatment of this topic is beyond the scope of this paper, we would like to sketch the basic ideas.

Our approach outlined so far yields a single pattern tree implementing a mapping of the form

$$f : (x_1, x_2, \ldots, x_n) \mapsto G(y) \ ,$$

where G is a fuzzy subset of the domain $\mathbb{Y}$ of the output variable Y. In the case of regression learning, it is indeed enough to consider a single fuzzy set like (1), since from this fuzzy set, the original value can be recovered through

$$y = G^{-1}(z) \ . \tag{7}$$

Yet, one may think of the following generalization of this approach: Suppose that the domain $\mathbb{Y}$ is discretized by means of a fuzzy partition consisting of fuzzy sets $G_1, G_2, \ldots, G_k$. Then, a single pattern tree model implementing a mapping

$$f_i : (x_1, x_2, \ldots, x_n) \mapsto G_i(y) \ ,$$

could be constructed for each of these fuzzy sets, either in a knowledge-driven way by hand, or in a data-driven way using the learning method proposed in Section 3. Given a vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, a corresponding ensemble of pattern trees produces a fuzzy description of the output in the form of a vector

$$\begin{aligned} f(\boldsymbol{x}) &= (f_1(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x})) \\ &= (z_1, \ldots, z_k) \in [0,1]^k \ . \end{aligned}$$

The simple decoding (7) then has to be replaced by a defuzzification step, which could be accomplished, for example, by

$$y^* = \arg\min_{y \in \mathbb{Y}} \|(z_1, \ldots, z_k) - (G_1(y), \ldots, G_k(y))\| \ .$$

Recall that the i-the pattern tree can be considered as a model describing conditions (on the input attributes) under which the output variable Y is in G_i, e.g., under which "Y is medium" or "Y is large". Interestingly, compared to rule-based fuzzy systems, the "direction" of modeling is thus reversed. In fact, in rule-based systems, one typically starts with the rule antecedents, i.e., one fixes conditions on the input attributes and then assigns a suitable (fuzzy) value for the output variable. In pattern trees, it is just the other way around: First, the (fuzzy) output values are fixed, and then conditions on the input attributes are specified.

Finally, it is worth mentioning that pattern tree-based fuzzy systems of the above kind are in a sense generalizations of (standard) rule-based systems, in which rule antecedents are combined by means of a t-norm and the rules themselves by means of a t-conorm. In fact, given a system of that kind, all rules with the same consequent part “Y is G_i” can be collected and represented as a two-level pattern tree: The first level consists of a node labeled with a t-conorm, while the second level consists of nodes labeled with a t-norm, one for each rule, aggregating the corresponding rule antecedents. Strictly speaking, a tree of that kind is not a proper pattern tree, since it is not binary. However, noticing that both t-norms and t-conorms are associative operators, it can easily be “binarized”.

5 Experiments

This section presents an experimental study that we conducted to get an idea of the performance of our pattern tree learner for regression.

5.1 Methods

We implemented our algorithms under the WEKA Machine Learning Framework [14]. For the experiments, we used the standard pattern tree learner for regression (PT-reg) as introduced in Section III. Moreover, we also included the “fuzzy systems” variant (PT-sys) outlined in Section IV, partitioning the domain of the output variable in a uniform way by means of three triangular fuzzy sets (the first with core $\{a\}$ and support (a, b), the second with core $\{b\}$ and support (a, c), and the third with core $\{c\}$ and support (b, c), where $[a, c]$ is the (observed) domain of the output and $b = (a + c)/2$).

For comparison, we included several other regression methods implemented in WEKA: Standard linear regression (LR), multilayer perceptrons (MLP), support vector machines with linear (SVM-lin) and RBF kernel (SVM-rbf), and regression trees (REPtree); all these algorithms are used with their default parameterization. Finally, we also included the fuzzy rule learner (FR) proposed in [13], which is implemented in the KEEL software [1].

5.2 Data Sets

The data sets used in the experiments are shown in Table 3. The table provides the number of instances (#instances), the number of numeric attributes (#num) and the number of nominal attributes (#nom) per data set, as well as the mean and standard deviation of the target attribute. The 12 data sets have been collected from the UCI [2] and the STATLIB [8] repositories.

5.3 Results

Table 4 provides a summary of the results in terms of the RMSE (root mean squared error), which has been obtained by averaging over 5 repetitions of a ten-fold cross validation. As can be seen, the pattern tree learners are quite competitive. In terms of average ranks, PT-reg is even the best method.

To test for statistical significance of the differences, we compared the methods in a pairwise way by means of a Wilcoxon-signed-rank test. It turns out, however, that the test fails to reject the null-hypothesis of equal performance most of the time, maybe due to the limited number of data sets included in this study. An exception is the fuzzy rule learner, which is significantly worse than both PT-reg and PT-sys (at a significance level of 5%). In any case, the results show that pattern tree learning is fully competitive to state-of-the-art regression methods in terms of predictive accuracy.

Table 3: Datasets with number of instances (#instances), numeric attributes (#num), nominal attributes (#nom) and mean (mean) and standard deviation (stddev) of target attribute.

	#instances	#num	#nom	mean	stddev
auto-mpg	390	8	0	23.42	7.81
concrete	1030	9	0	35.82	16.71
flare1M	323	8	3	0.14	0.48
flare2C	1066	8	3	0.3	0.84
forestfires	517	11	2	12.85	63.66
housing	506	14	0	22.53	9.2
imports-85	205	16	10	13207.13	7868.77
machine	209	7	2	105.62	160.83
servo	167	3	2	1.39	1.56
slump	103	11	0	36.04	7.84
winequality-red	1599	12	0	8.32	1.74
winequality-white	4898	12	0	5.88	0.89

Table 4: Experimental results in terms of RMSE. Additionally, for each data set, the rank of each method is show in brackets.

	LR	**MLP**	**SMO-lin**	**SMO-rbf**	**PT-reg**	**REPtree**	**PT-sys**	**FR**
auto-mpg	3.35(5)	3.24(3)	3.43(6)	3.54(7)	2.94(1)	3.34(4)	2.95(2)	5.87(8)
concrete	10.47(5)	8.01(3)	10.91(7)	10.8(6)	7.97(2)	7.21(1)	9.08(4)	13.7(8)
flare1M	0.44(1)	0.68(7)	0.46(5)	0.44(3)	0.47(6)	0.44(2)	0.45(4)	0.83(8)
flare2C	0.77(1)	0.99(7)	0.83(5)	0.83(6)	0.83(4)	0.77(2)	0.79(3)	1.15(8)
forestfires	47.6(4)	82.89(8)	46.56(2)	46.51(1)	61.67(6)	46.86(3)	52.61(5)	75.67(7)
housing	4.87(5)	4.09(1)	4.95(6)	5.5(7)	4.28(2)	4.56(3)	4.66(4)	7.43(8)
imports-85	2665.71(3)	2660.27(2)	2566.86(1)	2921.9(6)	2876(5)	3560.71(7)	2825.72(4)	8198.64(8)
machine	145.43(6)	160.43(8)	62.72(1)	79.2(5)	69.6(2)	147.82(7)	72.08(3)	78.23(4)
servo	1.11(5)	0.56(1)	1.27(7)	1.44(8)	0.68(2)	0.74(3)	0.76(4)	1.17(6)
slump	2.7(2)	0.67(1)	2.77(4)	4.94(6)	2.73(3)	5.04(7)	3.45(5)	7.41(8)
winequality-red	0.65(2)	0.73(7)	0.66(3)	0.66(4)	0.65(1)	0.68(6)	0.67(5)	1.13(8)
winequality-white	0.75(2)	0.79(6)	0.76(3)	0.76(5)	0.76(4)	0.75(1)	0.79(7)	1.3(8)
average rank	**3.42**	**4.50**	**4.17**	**5.33**	**3.17**	**3.83**	**4.17**	**7.42**

As an important advantage of pattern trees, especially in comparison to "black box" methods such as neural networks (MLP), we would like to highlight again their interpretability. As an illustration, Fig. 4 shows a pattern tree learned for the wine quality data set, where the problem consists of predicting the quality of wine on a scale from zero to ten,[2] given a number of properties such as concentration of alcohol, volatile acidity, sulphates, pH-value and several others. The tree can be interpreted in a quite

[2]Strictly speaking, this is an ordinal regression problem. However, due to the large number of levels, it seems legitimate (and perhaps even reasonable) to treat the ordinal scale as a numerical one.

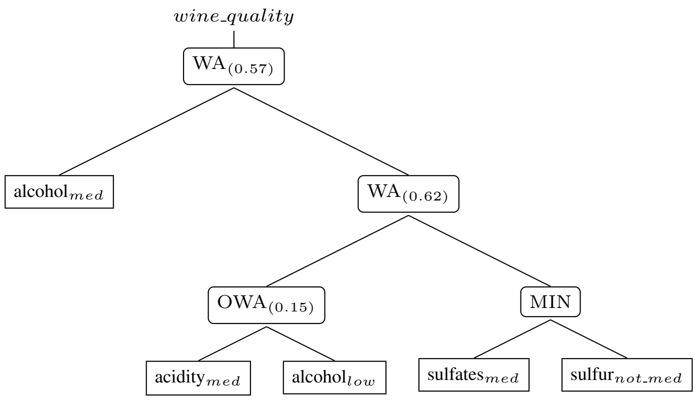

Figure 4: Pattern tree for red wine quality data (PT-reg).

intuitive way: The wine quality is an average of two criteria, namely a medium level of alcohol and a second criterion. This second criterion is in turn a complex one, namely an average of two sub-criteria. One of these sub-criteria is the conjunctive condition that the level of sulfates is medium and the level of sulfur not medium, and the second sub-criterion can be interpreted correspondingly.

Please note that, to get a basic understanding of the model, it is not necessary to look at the concrete operators and their parameterization. Instead, it is normally sufficient to distinguish between the main types of aggregation, namely conjunctive, disjunctive, and averaging. Upon closer inspection, the parameter values may of course provide useful extra information. For example, the weights in WA are in direct correspondence with the importance of the two criteria. Likewise, a disjunction (t-conorm) can be more or less compensating.

As a disadvantage of the pattern tree approach, we note that, at least in its current implementation, it has a high computational complexity. In fact, the addition of each single node of a pattern tree requires several passes over the training data, one for each candidate subtree. In this regard, it is to some extent comparable with the MLP, that is also trained in an iterative way. In terms of average runtime, however, we found that PT-reg is still more complex than MLP by a factor of about 3. Finding ways to reduce the runtime of pattern tree induction is therefore an important topic of future work.

Recall that, in the approach to fuzzy system modeling outlined in Section IV, one pattern tree is learned for each fuzzy term describing the output attribute. An example of such a model is given in Fig. 5, where we show the trees learned for the fuzzy outputs “high”, “medium”, and “low”, respectively, again for the wine quality data. These trees can be interpreted in a similar way as before: The first tree specifies properties under which the quality is high, the second one properties under which the quality is medium, and the third one properties under which the quality is low.

6 Concluding Remarks

In this paper, we have proposed a variant of pattern tree induction suitable for learning real-valued functions. Our experimental results on a number of benchmark data sets show that fuzzy pattern trees are able to approximate such functions in an accurate manner and, in fact, are competitive to state-of-the-art methods for regression in terms of predictive performance.

Not less importantly, pattern trees provide an interpretable model class that appears to very interesting from a modeling perspective. In fact, as argued in this paper, pattern trees constitute a viable alternative to classical fuzzy rule models, especially since their hierarchical structure allows for a more compact representation and for trading off accuracy against model simplicity in a seamless manner. In future work, we plan to elaborate on this aspect in more detail. Besides, as mentioned before, another important issue to be addressed in future work concerns the reduction of the runtime complexity of pattern tree induction.

A Java implementation of our PT-reg algorithm, running under the open-source machine learning framework WEKA, can be downloaded from the Internet.[3]

Acknowledgments

This research was supported by the German Research Foundation (DFG). The fuzzy rule learner used in the experiments was made available to us by the developers of the KEEL software (Alberto Fernández). We gratefully acknowledge this support.

References

[1] J. Alcalá-Fdez, L. Sánchez, S. Garcia, M.J. del Jesus, S. Ventura, J.M. Garrell, J. Otero, C. Romero, J. Bacardit, V.M. Rivas, J.C. Fernández, and F. Herrera. KEEL: A software tool to assess evolutionary algorithms to data mining problems. *Soft Computing*, 13(3):307–318, 2009.

[2] A. Asuncion and D. Newman. Uci machine learning repository, 2009. Accessed 13 Nov 2009.

[3] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth International Group, Belmont, CA, 1984.

[4] Zhiheng Huang, Tams D. Gedeon, and Masoud Nikravesh. Pattern trees induction: A new machine learning method. *IEEE Transactions on Fuzzy Systems*, 16(4):958–970, 2008.

[5] PJ. Huber. *Robust Statistics*. Wiley, 1981.

[6] J.S.R. Jang. Structure determination in fuzzy modeling: A fuzzy CART approach. In *Proceedings FUZZ-IEEE–94*, Orlando, 1994.

[3] `http://www.uni-marburg.de/fb12/kebi/research/`

[7] Erich Peter Klement, Radko Mesiar, and Endre Pap. *Triangular Norms*. Kluwer Academic Publishers, 2002.

[8] M. Meyer and P. Vlachos. Statlib data, software and news from the statistics community, 2009. Accessed 13 Nov 2009.

[9] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.

[10] T.L. Saaty. *The Analytic Hierarchy Process*. McGraw-Hill, 1980.

[11] B. Schweizer and A. Sklar. *Probabilistic Metric Spaces*. New York, 1983.

[12] R. Senge and E. Hüllermeier. Learning pattern tree classifiers using a co-evolutionary algorithm. In F. Hoffmann and E. Hüllermeier, editors, *Proceedings 19. Workshop Computational Intelligence*, pages 22–33, Dortmund, Germany, 2009. KIT Scientific Publishing.

[13] L.X. Wang and J.M. Mendel. Generating fuzzy rules by learning from examples. *IEEE Transactions on Systems, Man, and Cybernetics–Part C*, 22(6):1414–1427, 1992.

[14] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2 edition, 2005.

[15] R.R. Yager. On ordered weighted averaging aggregation operators in multi criteria decision making. *IEEE Transactions on Systems, Man and Cybernetics*, 18(1):183–190, 1988.

[16] Y. Yi, T. Fober, and E. Hüllermeier. Fuzzy operator trees for modeling rating functions. *International Journal of Computational Intelligence and Applications*, 8(4):413–428, 2009.

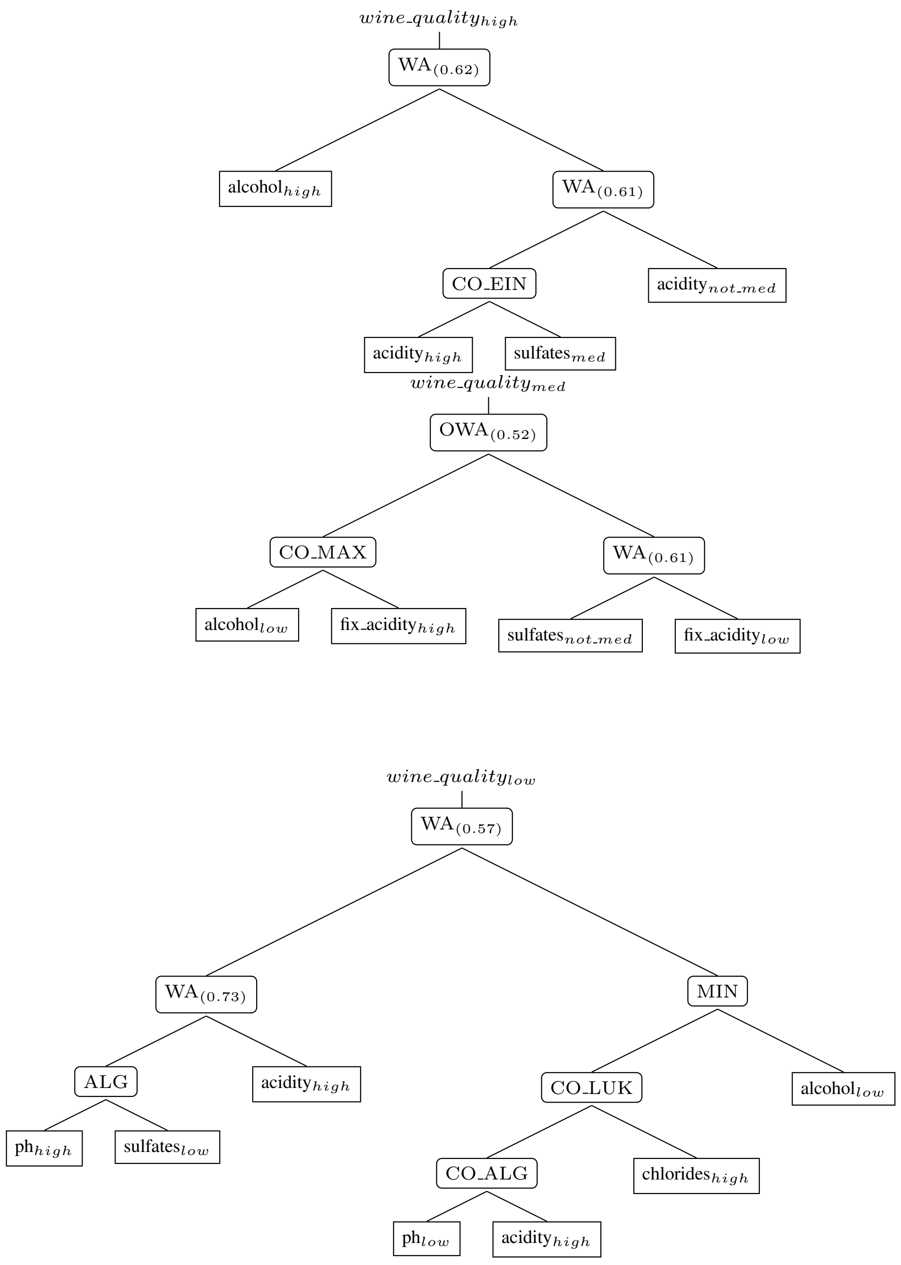

Figure 5: Pattern trees for the red wine quality data (PT-sys).