Inductive Learning of Fuzzy Regression Trees

Mario Drobics

Software Competence Center Hagenberg
4232 Hagenberg, AUSTRIA
mario.drobics@scch.at

Abstract

In this paper we present a novel approach to data-driven fuzzy modeling which aims to create highly accurate but also easily comprehensible models. This goal is obtained by defining a flexible but expressive language automatically from the data. This language is then used to inductively learn fuzzy regression trees from the data. Finally, we present a detailed comparison study on the performance of the proposed method and an outlook to future developments.

Keywords: Decision Tree Learning, Inductive Learning.

1 Introduction

Fuzzy logic based systems can be used to gain insights on a complex system for which no analytical model exists. For many complex technical applications the problem arises that no proper mathematical formulation can be found to describe the behavior of the according system. The only available information might be a set of measurements taken from the system. Then the goal is to find a function f that models the inherent connection between the input parameters (settings and measurements) and the goal parameter (final parameter of interest) that is hidden in the data.

To find such a function f, however, is not always the only objective. While statistical regression or neural networks allow to solve such kinds of machine learning problems, they leave the resulting function f as a *black box*, i.e. a plain function whose internals are difficult or impossible to comprehend. In many practical applications, however, qualitative insights into the structures of f are desirable. In the following, assume that we are not necessarily interested in the full function f, but at least in significant bits of knowledge about f and their inherent structures. Rule based systems or decision trees [4, 12] have been profen to be easily comprehensible and are therefor ideal for this task of qualitative *and* qualitative analysis.

For the remaining, let us consider a data set $\mathcal{X}$ of K samples $\mathcal{X} = \{\mathbf{x}^1, \ldots, \mathbf{x}^K\}$, where each sample $(i = 1, \ldots, K)$ has the same $(n+1)$-dimensional structure:

$$\mathbf{x}^i = (x_1^i, \ldots, x_n^i, x_{n+1}^i)$$
$$\in X_1 \times \cdots \times X_n \times X_{n+1} \qquad (1)$$

The first n dimensions/variables are the inputs; the last dimension/variable $n+1$ is the output under investigation. In the following, we refer to the r-th dimension $(r = 1, \ldots, n)$ as *r-th input attribute*. The $n+1$-th dimension is called *goal attribute*. Ideally, the overall objective of this machine learning problem is then to find a function

$$f : X_1 \times \cdots \times X_n \longrightarrow X_{n+1} \qquad (2)$$

such that the inherent connection between the input attributes and the goal attribute hidden in the data set $\mathcal{X}$ is modeled as well as possible.

To be able to handle numeric attributes in rule-based models, it is indispensable to define a discrete set of predicates for these kinds of attributes. If this quantization is done by means of partitions into crisp sets (intervals) as in traditional machine learning, small variations (e.g. noise) can cause

large changes in the model output and instable results. This entails the demand for admitting vagueness in the assignment of samples to predicates. Fuzzy sets [14] perfectly solve this problem of artificial preciseness arising from sharp interval boundaries.

A second benefit of fuzzy logic systems like fuzzy decision trees [15] or fuzzy rule-based methods [2] is, that they create not only a computational but also an interpretable model for f. It turned out, however, that in many cases the simple application of methods for creating interpretable, computational models from data is not sufficient. There is often the need for higher accuracy, by preserving the interpretability of the systems. Consequently, recently several approaches were developed to optimize given interpretable logic fuzzy systems [5]. These approaches, however, always focus on either interpretability *or* accuracy.

To overcome these limitations, we will compute semantically meaningful fuzzy sets a priori to the rule induction process, integrating user defined fuzzy predicates. We will then use these predicates for inductive learning of fuzzy decision trees to obtain comprehensible fuzzy models from data. To obtain models with higher accuracy a regularizes optimization technique can be applied to the whole model, afterward.

2 The Underlying Language

To define the underlying language for our fuzzy models, we have to consider the different types of input attributes that can occur. Basically, we can distinguish between three types of attributes:

Boolean categorical attributes: The domain X_i is an unstructured finite set of labels, for instance, types of car engines (gasoline, Diesel, hydrogen, electric) or classes of animals (birds, fish, mammals, etc.). The attribute values x_r^i are single elements of the label set X_i.

Fuzzy categorical attributes: There is again an unstructured finite set of labels, but with possible overlaps. Therefore, values of such kinds of variables may be fuzzy sets on this set of labels. For example, assume that we are given a finite set consisting of different grape varieties. Then blended wines (cuvees) cannot be assigned to single categories crisply.

Numerical attributes: The underlying domain X_i is the set of real numbers or a subset of these (e.g. an interval). The attribute values x_r^i are real numbers, e.g. pressures, temperatures, incomes, ratios, etc.

Fuzzy predicates for categorical attributes, boolean or fuzzy, can be defined easily in a straight forward manner. Finding appropriate fuzzy predicates for numerical attributes, however, is often a subtle problem for which different approaches exist.

In our approach, we create the fuzzy sets based on the data set given by considering the semantics of the corresponding linguistic expressions automatically using a method called *CompFS* [6]. By comprising also ordering based predicates we are able to define comprehensible, but still expressive predicates automatically [3].

3 Rule Induction

To create a decision or regression tree for a specific decision problem, *inductive learning* (i.e. learning from examples) is a widely used approach.

Recent approaches to apply fuzzy decision trees on regression problems try to create large trees which solve the resulting optimization problem [1, 9]. These solutions, however, can no longer be interpreted easily—which is usually one of the main advantages of regression trees over numerical optimization methods or artificial neural nets. Using pruning and back-fitting strategies can help to overcome this shortcoming [10]. All these approaches, however, tackle the problem of finding accurate but comprehensible models from an optimization point of view and do not pay attention to the underlying language used, nor are they capable of using a domain specific set of fuzzy predicates.

In our approach to inductive learning of fuzzy regression trees we pay special attention to comprehensibility, accepting a slightly lower performance

compared to other approaches. This is achieved by using the general language defined in section 2 and by creating as compact models as possible.

3.1 Fuzzy Regression Trees

A general regression tree consists of a *root node* with a number of *child nodes*. Each of these child nodes can either be a *leaf node* or the root node of a new subtree. If each non-leaf node has exactly two child nodes, the tree is called *binary*. We denote the set of all nodes with $\mathcal{N} = \{n^1, \ldots, n^N\}$, the set of all leaf nodes with $\mathcal{L} = \{n^{l_1}, \ldots, n^{l_L}\} \subset \mathcal{N}$ and the set of non-leaf nodes with $\mathcal{M} = \{n^{M_1}, \ldots, n^{M_N}\} \subset \mathcal{N}$ where we define the node n^1 to be the root node.

To each non-leaf node $n^i \in \mathcal{M}$, a predicate p^i is associated which is used to decide which of the child nodes to process next. For each non-leaf node $n^i \in \mathcal{M}$ the child nodes are denoted as n_1^i and n_2^i and we define that the left branch (n_1^i) is selected when the corresponding predicate p^i is fulfilled and the right one (n_2^i) otherwise. Each leaf node $n^j \in \mathcal{L}$ is associated with a constant value $c^j \in \mathbb{R}$ or a local model $c^j(\mathbf{x}), X \mapsto \mathbb{R}$.

3.2 Inductive Learning of Fuzzy Regression Trees—*FS-LiRT*

The basic idea behind *FS-LiRT* is to create a tree where the leaves approximate the desired goal function as good as possible. By associating numerical values (or functions) with the leave nodes, we finally obtain a Sugeno- or TSK-type controller. The method is called *FS-LiRT* (*F*uzzy *S*et based *Li*near *R*egression *T*rees) as in most cases linear models are associated with the leaf nodes.

We use the mean squared error measure which ensures that the model accuracy increases the larger the tree grows. The mean squared error for a given predicate P and a sample set $\mathcal{X}$ is computed according to:

$$\text{MSE}_{\text{DT}}(P, \mathcal{X}) = \frac{\sum_{\mathbf{x} \in \mathcal{X}} \mu_{\mathcal{X}}(\mathbf{x})(\tilde{z}_P(\mathbf{x}) - x_{n+1}))^2}{|\mathcal{X}|}, \quad (3)$$

$$\tilde{z}_P(\mathbf{x}) = t(P(\mathbf{x}))\bar{z}(\mathcal{X}|P) + t(\neg P(\mathbf{x}))\bar{z}(\mathcal{X}|\neg P),$$

where x_{n+1} is the desired goal value, $\tilde{z}_P(\mathbf{x})$ is an estimate of the output according to predicate P,

and

$$\bar{z}(\mathcal{X}|P) = \frac{\sum_{\mathbf{x} \in \mathcal{X}} t(P(\mathbf{x}))x_{n+1}}{\sum_{\mathbf{x} \in \mathcal{X}} t(P(\mathbf{x}))}$$

is the average goal value with respect to the predicate P.

Outline: *Starting with a single root node a binary regression tree is grown in a top-down manner. The mean squared error is computed and the predicates are sorted with respect to their actual relevance. Then the most relevant predicate is chosen and associated with the tree node under investigation. This procedure is repeated recursively until a stopping condition is fulfilled. Details are shown in Algorithm 1*

Algorithm 1 (FS-LiRT)

> **Input:** goal attribute m
> samples $X_{\text{cur}} = \{\mathbf{x}^1, \ldots, \mathbf{x}^K\}$
> set of test predicates $\mathcal{P}$
> **Output:** tree node N_{cur}
>
> **if** stopping criterion is fulfilled
> {
>   compute $c^{\text{cur}} = \{c_1^{\text{cur}}, \cdots, c_k^{\text{cur}}\}$
> N_{cur} is leaf node with class assignment C_{cur}
> }
> **else**
> {
>   find best predicate
> $P = \text{argmin}_{P \in \mathcal{P}} \text{MSE}_{\text{DT}}(P, X_{\text{cur}})$
>   compute new memberships for the left branch
> $\mu_{X^\oplus}(\mathbf{x}^i) = t(\mu_{X_{\text{cur}}}(\mathbf{x}^i) \wedge P(\mathbf{x}^i))$
>   compute left branch
> $N^\oplus = \mathbf{FS} - \mathbf{LiRT}(C, X^\oplus, \mathcal{P})$
>   compute new memberships for the right branch
> $\mu_{X^\ominus}(\mathbf{x}^i) = t(\mu_{X_{\text{cur}}}(\mathbf{x}^i) \wedge \neg P(\mathbf{x}^i))$
>   compute right branch
> $N^\ominus = \mathbf{FS} - \mathbf{LiRT}(C, X^\ominus, \mathcal{P})$
> N_{cur} is parent node with children $N^\oplus$ and $N^\ominus$
> }

The leaf node output c^j for a leaf node l^j is defined as the weighted average of the $n + 1$-th attribute (our goal attribute) according to:

$$c^j = \frac{\sum_{\mathbf{x} \in X} t(l^j(\mathbf{x}))x_{n+1}}{\sum_{\mathbf{x} \in X} t(l^j(\mathbf{x}))} \quad (4)$$

To achieve a more accurate approximation it is also possible to define the output c^j as a linear combination of the inputs. To preserve comprehensibility, we have to ensure that the linear terms are still interpretable. To do so, we do not use a simple linear combination of the input dimensions, but of a transposition with respect to the

center of the data under consideration according to

$$c^j(\mathbf{x}) = \alpha_0 + \sum_{l=1}^{n} \alpha_l(x_l - \bar{x}_l). \qquad (5)$$

$\bar{x}_l$ defines the mean of the samples in the l-th dimension according to b_l^j. Doing so, we can interpret α_0 as the output value in the "center" of the rule, and the α_l's as the change along the l-th dimension. By additionally applying a local variable selection, we are able to use local approximation functions which improve accuracy but are still comprehensible.

Finally, we have to solve the local least squares problem

$$\sum_{\mathbf{x} \in X} \left(t(n^j(\mathbf{x}))(x_{n+1} - \alpha_0^j - \sum_{l=1}^{n} \alpha_l^j(x_l - \bar{x}_l)) \right)^2 = \min_{\alpha_j},$$

where $\alpha^j = (\alpha_0^j, \alpha_1^j, \ldots, \alpha_{n-1}^j)^T$ are the according linear weights and $\bar{x}_l$ defines the weighted average of the samples under consideration with respect to the l-th dimension. As the corresponding system matrix is not guaranteed to have full rank, Tikhonov regularization is applied.

The algorithm stops if any of the following stopping criteria is fulfilled (Note that if pruning is applied, usually only the first stopping criteria is applied):

- No more samples: if the number of samples decreases under a certain threshold (Default: 10% of the orignal data).

- Minimum variance: if the variance of all samples in a node is below a given threshold (Default: 5% of the range of the goal attribute).

- Maximum depth reached: if the depth of the tree reaches a predefined maximum (Default: 10).

- No sufficient increase: if the relative increase with respect to the mean squared error (Default: 0.10).

Optionally, pruning can be applied to the tree generated by *FS-LiRT* to optimize the size of the tree. The goal is to achieve a good compromise between a models simplicity and its predictive accuracy, by removing irrelevant parts of the model. By pruning a tree, the new complexity of the model is automatically identified without the need to a priori establish thresholds for the stopping conditions which could be sensitive to problem specifics. Pruning also enhances the interpretability of a tree, a simpler tree being easier to interpret.

We use the same pruning technique as presented by [10]. They used a four step procedure, where first the inner nodes of the tree are sorted with respect to their sum-squared-error. Then a sequence of subtrees is generated by subsequently deleting the child nodes of the nodes in this sequence. Thirdly, the mean-absolute-error on a pruning data set is computed for each of these subtrees. Finally, the smallest tree within one standard error is selected.

3.3 Deduction with Fuzzy Regression Trees

To obtain real-valued output from fuzzy regression trees, the regression tree can be inferred directly by computing the output of a tree node $n^i(\mathbf{x})$ according to:

$$c^i(\mathbf{x}) = t(p^i(\mathbf{x}))c_1^i(\mathbf{x}) + t(\neg p^i(\mathbf{x}))c_2^i(\mathbf{x}) \qquad (6)$$

This formula is evaluated recursively to obtain the output of the root node c^1.

4 Examples

4.1 Two-Dimensional Example A

To illustrate the potential of the proposed method for fuzzy modeling, we tried to reconstruct the following function from data ($n = 2$, $X_1 = X_2 = [0, 100]$, $X_3 = [-100, 100]$):

$$f_{2D-A}(x_1, x_2) = x_2 \cdot \sin\left(\frac{2\pi x_1}{100}\right)$$

We selected $K = 1000$ random samples (x_1^i, x_2^i) from the range $X_1 \times X_2 = [0, 100]^2$. The final data set was constructed as

$$\mathcal{X} = \{(x_1^i, x_2^i, f_{2D-A}(x_1^i, x_2^i)) \mid i = 1, \ldots, K\}.$$

Six fuzzy sets with bell-shaped membership functions were created for the first input attribute x_1 and two for the second input attribute x_2.

FS-LiRT was executed to create a compact regression tree. In a second run, a larger tree was created and pruning was applied to remove unnecessary nodes. The pruned tree is shown in Fig. 1.

Parameter	Setting 1	Setting 2
t-norm	product	product
Fuzzy Sets	bell shaped	bell shaped
$supp_{min}$	0.05	0.01
$stddev_{min}$	0.1	0.001
$incr_{min}$	0.01	0.001
pruning	no	yes

Finally, we compared the output of the resulting fuzzy controller with the original values. The results for the original and the pruned tree are shown below. While the original tree was much larger than the pruned tree, the pruned tree shows a almost equal performance. Figure 2 shows plots of the original function f_{2D-A} and the function defined by the resulting Sugeno system of the pruned tree.

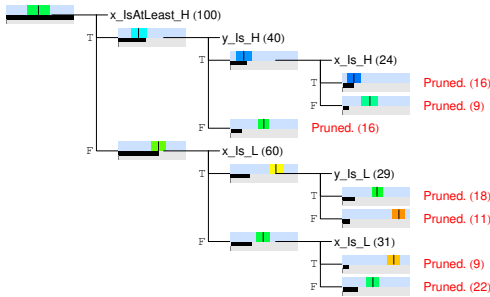| Parameter	Setting 1	Setting 2
Tree size	22	7
Correlation	0.97	0.98
Average Error	8.18	8.15
Average Squared Error	97.87	100.44

Figure 1: Regression tree constructed by FS-LiRT for the function approximation problem 2D-A after pruning

4.2 Comparison of Results

We have applied *FS-LiRT* using constant output functions *(FS-LiRT)* and linear output functions *(FS-LiRT Model)* on three regression problems from the UCI repository and on three artificial regression problem. We used 10-fold cross valida-
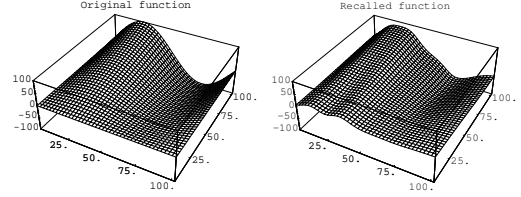

Figure 2: Test function f_{2D-A} (left) and the function defined by a Sugeno fuzzy system constructed by FS-LiRT (right)

tion and computed the average correlation coefficient. We compared the obtained results with two of our own methods, *FS-ID3* [7]—an other fuzzy decision tree learning method, and *FS-FOIL* [8]— a method for learning fuzzy rule bases. Furthermore we compared the results with three methods from the WEKA toolkit [13], namely *Linear-Regression*, *M5-Prime* [11], and *M5-Rules*. The first method creates a simple linear regression model to solve the regression learning problem. The latter two methods, *M5-Prime* and *M5-Rules* generate decision trees or decision rules. We ran these methods using the Weka Toolbox 3-4.

We compared the average correlation coefficient between the original output and the predicted values as well as the average model size. The results are shown in Figure 3. We can see, that for the data sets from the UCI repository our methods performed equally well as the other methods. For the complex two dimensional problems our methods create, however, much more compact models then the other methods.

5 Outlook

In this paper we have presented a novel algorithm for inductive learning of fuzzy decision trees which are both, comprehensible *and* accurate. By using local linear models based on a transformation of the original input dimensions we obtain interpretable models in the leafs, too.

Future work will focus on integrating cross validation in the optimization process. We hope, that this can help to further increase the stability of the models. Furthermore, we want to use this approach to automatically identify the regularization parameter of the Tikhonov regularization,

	AUTO-MPG	HOUSING	SERVO	2D-A	2D-B	6D
FS-ID3	0.9 (11)	0.87 (17)	0.87 (19)	0.93 (9)	0.74 (85)	0.92 (56)
FS-LIRT	0.91 (15)	0.91 (71)	0.94 (9)	0.93 (9)	0.84 (69)	0.93 (56)
FS-LIRT Model	0.92 (20)	0.91 (40)	0.96 (7)	0.97 (6)	0.88 (52)	0.99 (18)
FS-FOIL	0.86 (4)	0.83 (10)	0.92 (9)	0.88 (5)	-0.03 (1)	0.83 (7)
M5Rules	0.88 (12)	0.82 (13)	0.86 (5)	0.97 (86)	0.89 (29)	0.9 (51)
M5P	0.88 (17)	0.86 (21)	0.86 (10)	0.98 (153)	0.88 (100)	0.92 (150)
M5P-Modell	0.93 (4)	0.91 (14)	0.94 (6)	0.99 (102)	0.91 (112)	1. (18)

Figure 3: Comparison of average correlation coefficient and model size for different regression problems

which is crucial for semi-automatic application of the method.

Another future direction of research is the combination of different models into a combined model (some kind of additive regression). We have already made very promising experiments, which indicate that a combination of simple regression trees with constant output functions and simple regression models can be used to solve complex problems.

Acknowledgements

References

[1] J. F. Baldwin, J. Lawry, and T. P. Martin. A mass assignment based ID3 algorithm for decision tree induction. *Int. J. Intell. Syst.*, 12:523–552, 1997.

[2] P. Baranyi, Y. Yam, D. Tikk, and R. J. Patton. Trade-off between approximation accuracy and complexity: TS controller design via HOSVD based complexity minimization. In J. Casillas, O. Cordón, F. Herrera, and L. Magdalena, editors, *Interpretability Issues in Fuzzy Modeling*, volume 128 of *Studies in Fuzziness and Soft Computing*, pages 249–277. Springer, Berlin, 2003.

[3] U. Bodenhofer and P. Bauer. A formal model of interpretability of linguistic variables. In J. Casillas, O. Cordón, F. Herrera, and L. Magdalena, editors, *Interpretability Issues in Fuzzy Modeling*, volume 128 of *Studies in Fuzziness and Soft Computing*, pages 524–545. Springer, Berlin, 2003.

[4] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, editors. *Classification and Regression Trees*. CRC Press, 1984.

[5] J. Casillas, O. Cordón, F. Herrera, and L. Magdalena, editors. *Interpretability Issues in Fuzzy Modeling*, volume 128 of *Studies in Fuzziness and Soft Computing*. Springer, Berlin, 2003.

[6] M. Drobics. Choosing the best predicates for data-driven fuzzy modeling. In *Proc. 13th IEEE Int. Conf. on Fuzzy Systems*, pages 245–249, Budapest, July 2004.

[7] M. Drobics and U. Bodenhofer. Fuzzy modeling with decision trees. In *Proc. 2002 IEEE Int. Conf. on Systems, Man and Cybernetics*, volume 4, Hammamet, Tunisia, October 2002.

[8] M. Drobics, U. Bodenhofer, and E. P. Klement. FS-FOIL: An inductive learning method for extracting interpretable fuzzy descriptions. *Internat. J. Approx. Reason.*, 2002. (to appear).

[9] C. Z. Janikow. Fuzzy decision trees: Issues and methods. *IEEE Trans. Syst. Man Cybern. B*, 28(1):1–14, 1998.

[10] C. Olaru and L. Wehenkel. A complete fuzzy decision tree technique. *Fuzzy Sets and Systems*, 138(2):221–254, 2003.

[11] J. R. Quinlan. Learning with Continuous Classes. In *Proc. 5th Austr. Joint Conf. on Artificial Intelligence*, pages 343–348, 1992.

[12] J. R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[13] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann, 2000.

[14] L. A. Zadeh. Fuzzy sets. *Inf. Control*, 8:338–353, 1965.

[15] J. Zeidler and M. Schlosser. Continuous valued attributes in fuzzy decision trees. In *Proc. 8th Int. Conf. on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 395–400, 1996.