Soft Comput (2007) 11:421–438
DOI 10.1007/s00500-006-0107-1

FOCUS

Creating comprehensible regression models

Inductive learning and optimization of fuzzy regression trees using comprehensible fuzzy predicates

Mario Drobics · Johannes Himmelbauer

Published online: 7 October 2006

Abstract In this paper we will present a novel approach to data-driven fuzzy modeling which aims to create highly accurate but also easily comprehensible models. This is achieved by a three-stage approach which separates the definition of the underlying fuzzy sets, the learning of the initial fuzzy model, and finally a local or global optimization of the resulting model. The benefit of this approach is that it allows to use a language comprising of comprehensible fuzzy predicates and to incorporate expert knowledge by defining problem specific fuzzy predicates. Furthermore, we achieve highly accurate results by applying a regularized optimization technique.

Keywords Inductive learning · Fuzzy regression trees · Regularization · Interpretability

1 Introduction

Fuzzy logic based systems can be used to gain insights into 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 respective 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.

M. Drobics · J. Himmelbauer
Software Competence Center Hagenberg,
Hagenberg 4232, Austria
e-mail: mario.drobics@scch.at

J. Himmelbauer(✉)
e-mail: johannes.himmelbauer@scch.at

To find such a function f, however, is not always the only objective. While statistical regression (DraperSmith 1981) or neural networks (McClelland and Rumelhart 1986; Rumelhart and (McClelland 1986; Zurada 1992) 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. For such tasks, *rule-based systems* are most appropriate. They easily allow qualitative insight, since the function f is represented by logical rules in a close-to-natural-language manner. 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, i.e. *rules*.

For the remaining, let us consider a data set $\mathcal{X}$ of K samples

$$\mathcal{X} = \{\mathbf{x}^1, \ldots, \mathbf{x}^K\}, \tag{1}$$

where each sample $(i = 1, \ldots, K)$ has the same $(n+1)$-dimensional structure:

$$\begin{aligned}\mathbf{x}^i &= (x_1^i, \ldots, x_n^i, x_{n+1}^i)\\ &\in X_1 \times \cdots \times X_n \times X_{n+1}\end{aligned} \tag{2}$$

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 the *r-th input attribute*. The $n+1$-st dimension is called *goal attribute*. Ideally, the overall objective of this machine learning problem is then to find a function

$$f : X_1 \times \cdots \times X_n \longrightarrow X_{n+1} \tag{3}$$

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. Therefore, such machine learning problems can be regarded as a kind of data fitting.

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 classification quality and instable results. This entails the demand for admitting vagueness in the assignment of samples to predicates. Fuzzy sets (Zadeh 1965) perfectly solve this problem of artificial preciseness arising from sharp interval boundaries.

A second benefit of fuzzy logic systems like decision trees (Adamo 1980; Zeidler and Schlosser 1996) or rule-based methods (Baranyi et al. ; Mikut et al. 2005) is, that they create not only a computational but also an interpretable model for f. The resulting function helps the user to better understand the behavior of the system (Casillas et al. 2003). 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, while preserving the interpretability of the systems. Consequently, several approaches were developed recently to optimize existing interpretable fuzzy systems (Burger et al. 2002; Regattieri Delgodox et al. 2001).

In this paper we will present an approach to compute semantically meaningful fuzzy sets a priori to the rule induction process. We will then present an inductive learning algorithm for fuzzy decision trees which uses the so obtained fuzzy predicates to create comprehensible fuzzy models from data. Finally, we present an approach to optimize such (Takagi-)Sugeno fuzzy systems via linear approximation of the consequences. We pay special attention to the stability of the solution and to preserving the fuzzy system's interpretability. Applying variable selection is an additional contribution to reach this goal.

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 a 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 a 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 thereof (e.g. an interval). The attribute values x_r^i are real numbers, e.g. pressures, temperatures, incomes, ratios, etc.

Note that Boolean categorical attributes are special cases of fuzzy categorical attributes, since any crisp label can be considered as a fuzzy set of labels, too.

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.

The simplest approach is to create fuzzy sets which form a partition for each dimension and which are evenly distributed over the data range or which have the same cardinality (equi-distance binning, or equi-frequent binning). Although this approach is sufficient for basic calculations, it has strong limitations with respect to accuracy. Furthermore, it turns out that for unequally distributed data, such fuzzy sets often conflict with the user's intuition. To overcome these limitations, several approaches exist which try to fit the fuzzy sets to the given training data as well as possible. Most of these approaches use some kind of projection methods (Castellano et al. 2002; Klawonn and Kruse 1997) While fuzzy sets are typically defined over one dimension, fuzzy clustering methods can be used to identify higher-dimensional fuzzy sets (Hoeppner and Klawonn 2003 ; Yager and Filev 1994). Although the results of these two approaches are very promising, the resulting fuzzy sets do not always directly correspond to a linguistic expression – one of the main building blocks of fuzzy logic. Recent approaches in this direction use complex optimization techniques (Mikut et al. 2000) or create hierarchical fuzzy partitions Guillaume and Charnomordic (2004), requiring a high computational effort.

Alternatively, the fuzzy sets can also be computed ad hoc when creating the computational models. This approach has been applied in crisp algorithms like *CART* (Breiman et al.1984) and *C4.5* (Quinlan 1993) as well as in fuzzy logic based algorithms like *LoLiMoT* (Nelles et al. 2000) or fuzzy decision tree based methods (Baldwin et al. 1997; Janikow 1998; Zeidler and Schlosser 1996). Although this leads to very good numerical results, the drawback of these approaches is again a lack of linguistic expressions for the resulting fuzzy sets. The use of different but almost similar sets at different occurrences of an attribute makes it difficult to identify similarities between distinct parts of the rule base and complicates implementation of the obtained fuzzy models.

In contrast to the approaches mentioned above, we create the fuzzy sets based on the data set given by considering the semantics of the corresponding linguistic expressions. By comprising also ordering based predicates we are able to define comprehensible, but still expressive predicates automatically. Using this approach we can overcome the problem of identifying the ideal number of fuzzy sets in advance.

2.1 Generating meaningful fuzzy sets

To generate k unevenly distributed fuzzy sets according to the distribution of values in the data set $\mathcal{X}$, we developed a new algorithm called *CompFS* (Algorithm 1) (Drobics 2004). First, the centers c_i $(1 \leq i \leq k)$ of the fuzzy sets are initialized according to the data distribution. By initializing the fuzzy set centers with the according quantiles ($q_i = \frac{i-0.5}{k}$), we create an equi-frequent binning of the data set. To overcome problems which can occur when using the quantiles, it is necessary to readjust the fuzzy set centers by using a simple k-means algorithm. Of course, equal frequencies can no longer be guaranteed. Usually a few iterations of this clustering step suffice. The following example might illustrate this approach: Let our data set consists of $n/3$ zeros, $n/2$ fours, and n fives. When creating three fuzzy sets, the according quantiles (16, 50, and 83 %) are 0.5, and 5, respectively. This would result in two fuzzy sets having the same center. The data, however, consist of three well-separated groups. Applying the k-means algorithm rearranges the centers to 1, 4, and 5, which results in well-defined fuzzy sets. Note that using equi-distance binning would also fail to create useful fuzzy sets, as the middle fuzzy set would have no corresponding samples.

Finally, the fuzzy sets are computed around these centers. For piece-wise linear fuzzy sets, the i-th set $(1 < i < k)$ is defined by its support points $s_{(i,j)}$ which are defined according to

$$\begin{aligned} s_{(i,1)} &= \left(c_i - (1+o)\cdot\frac{(c_i - c_{i-1})}{2}, 0\right) \\ s_{(i,2)} &= \left(c_i - (1-o)\cdot\frac{(c_i - c_{i-1})}{2}, 1\right) \\ s_{(i,3)} &= \left(c_i + (1-o)\cdot\frac{(c_{i+1} - c_i)}{2}, 1\right) \\ s_{(i,4)} &= \left(c_i + (1+o)\cdot\frac{(c_{i+1} - c_i)}{2}, 0\right), \end{aligned} \qquad (4)$$

where o is the percentage of overlap between two sets which has to be specified in advance. Bell shaped fuzzy sets are directly defined by their membership function

$$\mu_{s_i}(x) = \mathrm{e}^{-\frac{1}{2}\left(\frac{c_i - x}{w}\right)^2}, \qquad (5)$$

where $w = (c_{i+1} - c_{i-1})/4$. The resulting families of fuzzy sets form a partition and are in a proper order – to ensure highest interpretability of the results (Bodenhofer and Bauer 2003).

To eliminate outliers, we use the 2 and 98% quantiles of the data as the upper and lower bound of the resulting partition, respectively. Although the complexity of computing the quantiles and the clustering is at most $\mathcal{O}(n\log(n))$, for large data sets these computations may be performed on a (representative) subset of the original data.

Algorithm 1 (CompFS)

```
Input:    samples $\mathcal{X} = \{\mathbf{x}^1, \ldots, \mathbf{x}^K\}$
          number of fuzzy sets to compute $k$
          data dimension $d$
Output:   fuzzy sets $\mathcal{S}$

initialize centers $c_i = \text{quantile}_{\frac{i-1/2}{k}}(X_d),\ i \in 1, \ldots, k$
do {
    compute new centers $\bar{c}_i$ as the average of all samples
        which are closest to center $c_i$.
    $\delta = \frac{1}{k} \cdot \sum_{i=1}^{k} |\bar{c}_i - c_i|$
    $c_i = \bar{c}_i,\ i = 1, \ldots, k$
} while $\delta > 0.01$
compute the fuzzy sets $\mathcal{S}$ around the centers $c_i$
    according to (4)
```

Figure 1 shows some examples of how the fuzzy sets are computed for different data distributions. In the first example, normally distributed data was used. The resulting fuzzy sets have almost similar cardinality and can be easily identified with linguistic expressions ranging from *very small* to *very high*. In the data set used in the the second example, the density decreases as the values increase. Therefore, the left most fuzzy sets are very narrow, while the other fuzzy sets are stretched to have a higher cardinality. In the third example, the data consists of two diverse normal distributions. To compensate for the resulting gap in the data, the inner two fuzzy sets are stretched to the center. In the fourth example, the data set is distributed very unevenly. By using *CompFS*,

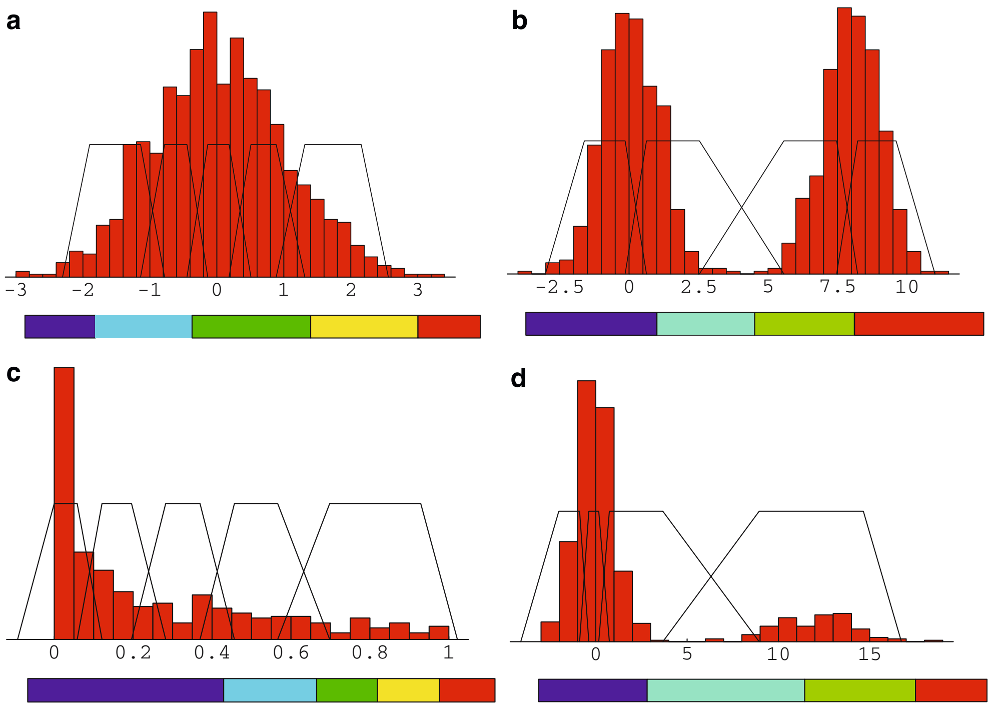


Fig. 1 Fuzzy sets created using CompFS

it was possible to identify fuzzy sets with almost equal cardinality which perfectly fit the given data distribution automatically.

Although *CompFS* is capable of computing the fuzzy sets automatically with respect to a given data distribution, it requires the actual number of fuzzy sets as an input. In our experiments it has, however, turned out that the actual choice of this number influences the performance of the resulting model only slightly as long as a sufficiently large number of fuzzy sets is created. This is achieved by using ordering-based predicates and by defining the underlying fuzzy sets according to the actual distribution of the data such that there is a good chance, that a sufficiently good split is found already with a low number of sets. A detailed discussion of our experiments can be found in Sect. 5.1.

2.2 Defining appropriate predicates

After defining the underlying fuzzy sets, it is necessary to define appropriate predicates using these fuzzy sets. A fuzzy predicate p on X is uniquely represented by its truth function

$$t(p) : X \to [0,1] \tag{6}$$

where $t(p(x))$ is interpreted as the degree to which the value x fulfills the predicate p.

Suppose that the r-th attribute is numerical. This means that $X_r \subseteq \mathbb{R}$ and the values in the r-th component are real numbers. We assume that, for attribute r, a family of N_r linguistic labels $M_{r,1}, \ldots, M_{r,N_r}$ is defined. Depending on the underlying context of the attribute under consideration, these labels can be natural language expressions like *very low*, *medium*, and *large*. To each label $M_{r,j}$, we assign a fuzzy set with membership function $\mu_{M_{r,j}} \in \mathcal{F}(X_r)$ $(j = 1, \ldots, N_r)$ using *CompFS*. Furthermore, we can define the complement and the smallest superset with non-decreasing/non-increasing membership function for these sets. These new fuzzy sets correspond to the linguistic expressions *is not*, *is at least*, and *is at most*, respectively (Fig. 2).

Given a set of linguistic labels $M_{r,1}, \ldots, M_{r,N_r}$ and their corresponding semantics modeled by fuzzy sets, we can now define $4 \cdot N_r$ atomic fuzzy predicates. The degrees to which a sample $\mathbf{x} \in X_1 \times \cdots \times X_{n+m}$ fulfills these predicates can be computed as follows ($j = 1, \ldots, N_r$):

$$\begin{aligned} t(\mathbf{x} \text{ is } M_{r,j}) &= \mu_{M_{r,j}}(x_r) \\ t(\mathbf{x} \text{ is not } M_{r,j}) &= 1 - \mu_{M_{r,j}}(x_r) \\ t(\mathbf{x} \text{ is at least } M_{r,j}) &= \sup\{\mu_{M_{r,j}}(u) \mid u \leq x_r\} \\ t(\mathbf{x} \text{ is at most } M_{r,j}) &= \sup\{\mu_{M_{r,j}}(u) \mid u \geq x_r\} \end{aligned} \tag{7}$$

Although the two latter ordering-based predicates are not absolutely necessary, they help to improve compactness, expressiveness, and interpretability of the results

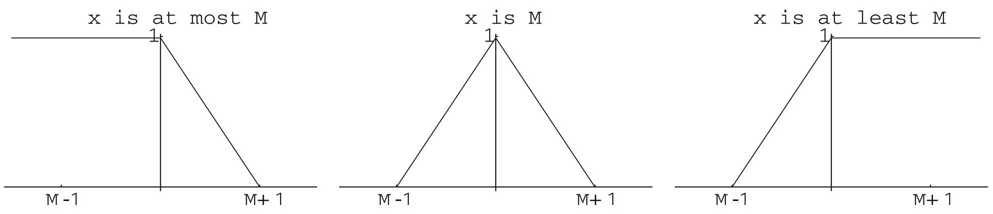


Fig. 2 Fuzzy predicates

(Bodenhofer 1999a,b; Bodenhofer and Bauer 2003; DeCock et al. 2000).

3 Rule induction

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

Using not only crisp but also fuzzy predicates, decision trees can be used to model vague decisions. Several approaches dealing with such fuzzy decision trees focus on the problem of vague class memberships (Maher and Clair 1993; Marsala 2000; Peng and Flach 2001; Wang et al. 2003; Yuan and Shaw 1995; Zeidler and Schlosser 1996) Viewing decision trees as a compressed representation of a (fuzzy) rule set enables us to use decision trees not only for classification, but also for approximation of continuous output functions. Recent approaches in this direction try to create large trees that solve the resulting optimization problem (Baldwin et al. 1997; Janikow 1998). 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 (Olaru and Wehenkel 2003). All these approaches, however, tackle the problem of finding accurate yet still 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. This is achieved by using the general language defined in Sect. 2 and by creating models which are as compact as possible. In contrast to most existing approaches, the trees generated using our method can be directly transformed to a corresponding TSK fuzzy system. These models are finally tuned with respect to their accuracy using a local or post-optimization technique described in Sect. 4. By introducing a novel transformation on the rule consequences we are able to achieve numerical accuracy without weakening the systems comprehensibility.

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 inner 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 inner 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 ease notation we will furthermore denote the index set of all leaf nodes with $L = \{l_1, \ldots, l_L\}$.

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 inner node $n^i \in \mathcal{M}$ the child nodes are denoted as n^i_1 and n^i_2, and we define that the left branch (n^i_1) is selected when the corresponding predicate p^i is fulfilled and the right one (n^i_2) otherwise. The uniquely determined path from the root node n^1 to a sub-node $n^j \in \mathcal{N}$ is called *complete branch* of the node and will be denoted as b^j. Each leaf node $n^j \in \mathcal{L}$ is associated with a constant value $c^j \in \mathbb{R}$ or a local model $c^j(\mathbf{x}), X \mapsto \mathbb{R}$.

To ensure comprehensibility of the local linear models, we do not use a simple linear combination of the input dimensions,

$$c^j(\mathbf{x}) = \alpha_0 + \sum_{l=1}^{n} \alpha_l x_l,$$

but of a reformulation with respect to the center of the data under consideration according to

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

$\bar{x}_l$ defines the mean of the samples in the l-th dimension according to b^j_l. This eases interpretation as in contrast to usual TSK models, the rule output can be interpreted easily. Actually, we can interpret α_0 as the mean output value of all data points the rule applies to. The $\bar{x}_l$'s then characterize this mean with respect to the l's dimension. The α_l's finally describe the influence of the l's dimension when x_l is below or above $\bar{x}_l$. The α_l's might also be normalized (e.g. to a standard deviation of 1 in each dimension) to obtain an ordered list of influencing factors. For a rough output of the model only α_0 might be displayed. If more detailed information is needed, the k top ranked α_l's can then be displayed.

In the following, we will restrict ourselves to binary regression trees. We overcome the main problem of binary trees – their increasing size for complex problems – by using the flexible underlying language, especially ordering-based predicates as defined in Sect. 2. This enables us to determine the ideal segmentation point automatically and to reduce the overall number of predicates involved.

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 well as possible. By associating numerical values (or functions) with the leaf 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.
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.*

To decide for each node which predicate to take, we use the mean squared error measure which ensures that the model accuracy increases as the tree grows. The mean squared error for a given predicate P and a sample set $\mathcal{X}$ is computed according to

$$\mathrm{MSE}_{\mathrm{DT}}(P,\mathcal{X}) = \frac{\sum_{\mathbf{x}\in\mathcal{X}} \mu_{\mathcal{X}}(\mathbf{x})(\tilde{z}_P(\mathbf{x}) - x_{n+1})^2}{|\mathcal{X}|}, \tag{8}$$

$$\tilde{z}_P(\mathbf{x}) = t(P(\mathbf{x}))\bar{z}(\mathcal{X}|P) + t(\neg P(\mathbf{x}))\bar{z}(\mathcal{X}|\neg P), \tag{9}$$

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

Algorithm 2 (FS-LiRT)

```
Input:    goal attribute $m$
          samples $X_{\mathrm{cur}} = \{\mathbf{x}^1, \dots, \mathbf{x}^K\}$
          set of test predicates $\mathcal{P}$
Output:   tree node $N_{\mathrm{cur}}$

if stopping criterion is fulfilled
{
    compute $c^{\mathrm{cur}} = \{c_1^{\mathrm{cur}}, \cdots, c_k^{\mathrm{cur}}\}$
    $N$cur is leaf node with class assignment $C_{\mathrm{cur}}$
}
else
{
    find best predicate $P = \mathrm{argmin}_{P\in\mathcal{P}} \mathrm{MSE}_{\mathrm{DT}}(P, X_{\mathrm{cur}})$
    compute new memberships for the left branch
        $\mu_{X^\oplus}(\mathbf{x}^i) = t(\mu_{X_{\mathrm{cur}}}(\mathbf{x}^i) \wedge P(\mathbf{x}^i))$
    compute left branch
        $N^\oplus = \mathbf{FS-LiRT}(C, X^\oplus, \mathcal{P})$
    compute new memberships for the right branch
        $\mu_{X^\ominus}(\mathbf{x}^i) = t(\mu_{X_{\mathrm{cur}}}(\mathbf{x}^i) \wedge \neg P(\mathbf{x}^i))$
    compute right branch
        $N^\ominus = \mathbf{FS-LiRT}(C, X^\ominus, \mathcal{P})$
    $N_{\mathrm{cur}}$ is parent node with children $N^\oplus$ and $N^\ominus$
}
```

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

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

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

$$c^j = \frac{\sum_{\mathbf{x}\in X} t(n^j(\mathbf{x}))x_{n+1}}{\sum_{\mathbf{x}\in X} t(n^j(\mathbf{x}))} \tag{10}$$

To achieve a more accurate approximation it is also possible to define the output c^j as a linear combination of the inputs. This can be achieved by solving the local least squares problem

$$\sum_{\mathbf{x}\in X} \Big(t(n^j(\mathbf{x}))\big(x_{n+1} - \alpha_0^j - \sum_{l=1}^{n} \alpha_l^j(x_l - \bar{x}_l)\big)\Big)^2 = \min_{\boldsymbol{\alpha}_j}, \tag{11}$$

where $\boldsymbol{\alpha}^j = (\alpha_0^j, \alpha_1^j, \dots, \alpha_n^j)^T$ 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. It has, however, to be noted that this optimization problem can only be solved if the according observation matrix has full rank. This motivates the usage of regularized optimization techniques to find optimal values for the α-s. This approach is described in more detail in Sect. 4.2.

The algorithm stops if any of the following stopping criteria is fulfilled:

- No more samples: if the number of samples decreases under a certain threshold (Default: 10% of the original 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

$$\frac{\mathrm{MSE}_{\mathrm{DT}}(p^i, \mathcal{X}^i) - \mathrm{MSE}_{\mathrm{DT}}(p^j, \mathcal{X}^j)}{\mathrm{MSE}_{\mathrm{DT}}(p^i, \mathcal{X}^i)}$$

  (c^j being a child node of c^i) is below a given threshold (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 model's simplicity and its predictive accuracy by removing irrelevant parts of the model. If pruning is applied, only the first stopping criterion is used. 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 the data set under investigation. This facilitates the application of the method also for non-experts and allows us to integrate this method into a semi-automatic problem solver.

We use the same pruning technique as presented by (Olaru and Wehenkel 2003). They used a four-step procedure, where first the inner nodes of the tree are sorted with respect to their summed 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 decision trees

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

$$c^i(\mathbf{x}) = t(p^i(\mathbf{x}))c_1^i(\mathbf{x}) + t(\neg p^i(\mathbf{x}))c_2^i(\mathbf{x}) \qquad (12)$$

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

Alternatively, we can transform the regression tree to a corresponding (Takagi-)Sugeno fuzzy system (TSK system) (Takagi and Sugeno 1985). The transformation is performed according to:

IF $n^j(\mathbf{x})$ THEN $c^j(\mathbf{x})$, $\quad \forall j \in L.$

The degree $n^j(\mathbf{x})$ to which a sample $\mathbf{x}$ belongs to the leaf node n^j is computed as the conjunction of the predicates on the corresponding complete branch b^j:

$$n^j(\mathbf{x}) = \bigwedge_{p^i \in b^j} b_i^j(p^i(\mathbf{x})) \qquad (13)$$

The function $b_i^j(p^i)$ returns p^i when the left (true) branch and $\neg p^i$ when the right (false) branch of node n^i is part of b^j.

Such a TSK system is defined as a mapping which assigns to each observation a crisp output value of the real domain. In the case of $\sum n^j(\mathbf{x}) > 0$ for all $\mathbf{x} \in X$, the input-output function $F_s : X \to X_{n+1}$ associated to Eq. (13) can be given explicitly by

$$F^j(\mathbf{x}) = \frac{\sum_{j \in L} n^j(\mathbf{x})\, c^j(\mathbf{x})}{\sum_{j \in L} n^j(\mathbf{x})}. \qquad (14)$$

In the following we will show a way to optimize a given TSK fuzzy system of the form Eq. (14) by tuning the rules' consequences $c^j(\mathbf{x})$.

4 Post-optimization of fuzzy rule bases

In the previous two chapters we presented a method for creating easily comprehensible fuzzy models. The trade off for obtaining higher comprehensibility is a lower accuracy, as smaller models have to involve more general rules/nodes. In our approach we overcome this drawback by adding more expressive output functions to the leaf nodes [as in Eq. (11)] or by performing post-optimization of the complete fuzzy system. In (Burger et al. 2002), a combination of both is presented, where—besides the coefficients of the affine linear consequences c^j – the underlying fuzzy sets themselves are fitted by optimizing the positions of the fuzzy sets' interpolation points. This approach results in a nonlinear optimization problem. Here, however, we decided to restrict ourselves to the optimization of the consequences, which leads to a linear least squares problem which can be solved easier and faster. Moreover, by doing so we avoid the danger of ending up in degenerated fuzzy sets which are no longer comprehensible.

4.1 Derivation of the least squares system

For the case that we want to perform a local optimization of the output functions we have already defined the least squares system in Eq. (11). For the global case, however, defining the according system is a little bit more complex.

In the following we will assume that the fuzzy rules fulfill the partition of unity, i.e. $\sum_{j\in L} n^j(\mathbf{x}) = 1$ for all $\mathbf{x} \in X$. To ease notation, we first restrict ourselves to the one-dimensional case, i.e. a single-input single-output system.

Under these assumptions, tuning a TSK system globally reduces to fitting a set of data $\mathcal{X}$ by a linear combination of the rules' membership functions in the least squares sense, i.e. seeking a solution of the minimization problem,

$$\sum_{\mathbf{x}\in\mathcal{X}} \left(x_{n+1} - \sum_{j\in L} n^j(\mathbf{x}) \cdot (\alpha_0^j - \alpha_1^j(x_1 - \bar{x}_1)) \right)^2 = \min_{\alpha},$$

or written in matrix form

$$\frac{1}{2}\|\mathbf{x}_{n+1} - N^{\text{obs}}\alpha\|^2 = \min_{\alpha}, \tag{15}$$

where $N^{\text{obs}} = \left(N \mid N_{Ap}\right)$ represents the observation matrix with

$$N = \begin{pmatrix} n^1(\mathbf{x}^1) & \cdots & n^{l_L}(\mathbf{x}^1) \\ \vdots & \ddots & \vdots \\ n^1(\mathbf{x}^K) & \cdots & n^{l_L}(\mathbf{x}^K) \end{pmatrix}$$

and

$$N_{Ap} = \begin{pmatrix} n^1(\mathbf{x}^1)(x_1^1 - \bar{x}_1) & \ldots & n^{l_L}(\mathbf{x}^1)(x_1^1 - \bar{x}_1) \\ \vdots & \ddots & \vdots \\ n^1(\mathbf{x}^K)(x_1^K - \bar{x}_1) & \ldots & n^{l_L}(\mathbf{x}^K)(x_1^K - \bar{x}_1) \end{pmatrix}$$

$\mathbf{x}_{n+1} := (x_{n+1}^1, \ldots, x_{n+1}^K)$ defines the goal vector of the system. As the solution of problem Eq. (15), we obtain the vector $\alpha = (\alpha_0^1, \ldots, \alpha_0^{l_L}, \alpha_1^1, \ldots, \alpha_1^{l_L})$, containing the coefficients of the approximation function for each rule.

So far we have treated only the case of a system with one-dimensional input. In many applications, however, we have to deal with large numbers of input variables. This extension to the multi-dimensional situation can be obtained without many modifications (except with respect to notation). For each additional input dimension we only need to add l_L columns to the matrix N_{Ap}. The observation matrix finally has dimension $K \times l_L(n+1)$ for the global case. For the local case we obtain a $\tilde{K} \times (n+1)$ dimensional observation matrix, where $\tilde{K}$ is the number of samples with a degree of fulfillment greater than zero.

4.2 Tikhonov-regularization of the system

In Sect. 3, we arrived at a linear least squares problem with – in the global case – a $K \times (l_L(n+1))$ observation matrix N^{obs}.

Equation (15) has a unique solution if and only if the observation matrix N^{obs} has full rank, which is equivalent to the – in approximation theory well-known — Schoenberg-Whitney condition (DeBoor 1978). In the case of full rank, we achieve the unique solution by solving the linear system $(N^{\text{obs}})^T N^{\text{obs}}\alpha = (N^{\text{obs}})^T \mathbf{x}_{n+1}$. In the case of a rank-deficient matrix (i.e., $r := \text{rank}(N^{\text{obs}}) < l_L(n+1)$), the least squares problem (15) is no longer uniquely solvable. The set of solutions consists of the linear manifold

$$\mathbf{x}^\dagger + N(N^{\text{obs}}).$$

$\mathbf{x}^\dagger$ denotes the unique least squares solution of minimal (Euclidean) norm, given by $\mathbf{x}^\dagger = (N^{\text{obs}})^\dagger \mathbf{x}_{n+1}$ (where $(N^{\text{obs}})^\dagger$ is the Moore–Penrose inverse or pseudo-inverse) and $N(N^{\text{obs}})$ denotes the nullspace of N^{obs} with dimension $(l_L(n+1)) - r$. Numerical calculation of the Moore-Penrose inverse, however, belongs to the class of so-called *ill-posed problems*. It can be shown that small perturbations in the system matrix can cause arbitrarily great differences in the result of the inverses.

For our application of optimizing a fuzzy rule base, we cannot guarantee an observation matrix N^{obs} with full rank, as the rank of the matrix is dependent on the already given and fixed rule conditions and most of all on the given data set. With an increasing ratio of number of rules l_L and input dimension n to the number of data samples K, the probability of ending up in a system with rank deficient N^{obs} rises. As, in our case, we have to deal with inexact input data (e.g. measurement errors) as well as with round-off errors during numerical computation, we cannot be sure of always obtaining a stable solution to our problem. Therefore, we have to use regularization techniques where the *ill-posed problem* is replaced by a family of similar, well-posed problems.

We will see that regularization of the problem has also positive effects on the condition number of the observation matrix N^{obs} and consequently of $(N^{\text{obs}})^T N^{\text{obs}}$, the matrix to be inverted for solving Eq. (15). The condition number of a matrix A is defined as $\kappa(A) := \|A\|\|A^{-1}\|$ and is a measure of a linear system's sensibility towards noisy input data. The higher κ, the higher is also the error of the solution with respect to the input error. The relative error of the solution of $A(x+u) = y + v$ can be estimated by

$$\frac{\|u\|}{\|x\|} \leq \|A\|\|A^{-1}\|\frac{\|v\|}{\|y\|}.$$

A similar error estimation exists for noisy system matrices $A + S$. Hence the condition number κ is a decisive measure of how well the numerical solution will behave. As we will see later, ill-conditioned systems (i.e. systems with very high κ) are generated in many cases of optimizing fuzzy rule bases.

To guarantee a unique and stable solution, we will now apply a regularization method on the given system. In Burger et al. (2002), a numerical analysis of Smoothing and Tikhonov regularization was made. In the following we will apply the Tikhonov method to our system Eq. (15). In our case, this amounts to simply adding to the cost function of the given system Eq. (15) the squared sum of the coefficients α_l^j (weighted by a *regularization parameter* β) as a penalty term.

Applying this regularization method we finally arrive at the following observation matrix

$$N^{\text{reg}} = \begin{pmatrix} N & \vert N_{Ap} \\ \sqrt{\beta} I & \end{pmatrix} \tag{16}$$

and output vector

$$\alpha = (\alpha_0^1, \ldots, \alpha_0^{l_L}, \ldots, \alpha_n^1, \ldots, \alpha_n^{l_L})^T. \tag{17}$$

for Eq. (15). The resulting least squares problem can be written as

$$\sum_{\mathbf{x} \in \mathcal{X}} \left(x_{n+1} - \sum_{j \in L} c^j(\mathbf{x}) n^j(\mathbf{x}) \right)^2 + \beta \sum_{k=1}^{l_L(n+1)} \alpha_k^2 = \min_{\alpha}, \tag{18}$$

where $c^j(\mathbf{x}) := \alpha_0^j + \alpha_1^j (x_1 - \bar{x}_1) + \ldots + \alpha_n^j (x_n - \bar{x}_n)$ for $j \in L$.

The penalty term $\beta \sum_{k=1}^{l_L(n+1)} \alpha_k^2$ causes the absolute values of the coefficient to be kept as low as possible. The higher the regularization parameter β is chosen, the closer to zero the coefficients are kept and consequently, the more we will achieve a stable, but also less accurate solution of our problem. In fact, it turns out that the choice of β is a crucial task; one has to find a good compromise between accuracy and stability.

4.3 Variable selection via forward selection

As already mentioned, we want to use our method for problems with large input dimensions. Hence, taking into consideration that for each rule we have a high-dimensional linear output-function $c^j(\mathbf{x})$, we end up in a high-dimensional observation matrix that is not sparse. Variable selection, i.e. only using a subset of all input parameters for each approximation $c^j(\mathbf{x})$, is a good tool to alleviate this drawback caused by high-dimensional input data.

There are three reasons why additional variable selection is recommendable. First, a system matrix approximation with all variables involves many entries, whereas our wish is to achieve a matrix which is as sparse as possible. Second, a very important criterion for a fuzzy system is its interpretability. A system that uses only the most "decisive and important" features for the output of each rule, is clearly expected to be more "readable" and interpretable than one using full selection. And thirdly, the danger of overfitting the data (most of all in case of a high ratio of the number of input parameters to the number of samples) is expected to decrease.

The way how we use this method for global post-optimization is as follows: First we apply variable selection to each rule separately. That is, we take one rule after the other and identify the most significant variables for the corresponding "one-rule-system". Then we optimize the global fuzzy system, using – for each output function – only the variables selected at the corresponding rule.

Looking at the linear approximation function (for the i-th test-point)

$$\hat{y}^i = \theta_0 + \theta_1 x_1^i + \theta_2 x_2^i + \cdots + \theta_n x_n^i, \tag{19}$$

where the x_k is called regressor and θ_k the regression coefficient, we must first determine an error measure with respect to which we want to find optimal parameters for θ_k. Choosing the sum of squares of the distances between the measured values x_{n+1}^i and the estimated values $\hat{y}^i$,

$$\sum_{i=1}^{K} (x_{n+1}^i - \hat{y}^i)^2 \tag{20}$$

the optimal parameters can be calculated by

$$\underline{\hat{\theta}} = \left(\underline{X}^T \underline{X} \right)^{-1} \underline{X}^T \underline{y}, \tag{21}$$

if the matrix $\underline{X}^T \underline{X}$ is regular with $\underline{X} = (\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_n)$, $\underline{x}_j = (x_j^1, \ldots, x_j^K)$ and $\underline{y} = (x_{n+1}^1, \ldots, x_{n+1}^K)$.

4.3.1 Forward selection

The next step is to select the most significant and decisive variables. We will use a simplified *forward selection* as we have to run this algorithm for each rule. The structure of this algorithm is the following:

1. For each regressor candidate x_k: Calculate the solution of the one-dimensional regression with regressor x_k and output variable x_{n+1}.
2. Determine the best-fitting, most significant solution (using the R^2 measure). Let us denote the corre-

sponding regressor with $\underline{x}_A$ and the associated parameter with $\hat{\theta}_{A0}$ for the constant part and $\hat{\theta}_A$ for the linear part.
3. Modify the output variable: $\underline{y}_A = \underline{y} - (\hat{\theta}_{A0} + \hat{\theta}_A \underline{x}_A)$ (i.e.: $\underline{y}_A$ is the part of $\underline{y}$ that cannot be described linearly via $\underline{x}_A$).
4. For all remaining regressor candidates x_k: Repeat step 1–3 with the new output variable $\underline{y}_A$.
5. Repeat this procedure (1–4) until a given number n_s of regressors is selected or a certain approximation quality criterion is fulfilled.

We chose this procedure as it is fast, as in the i-th step only $n - i - 1$ one-dimensional regression systems have to be solved. Further performance improvements are achieved by applying this procedure only on those samples for which the according rule j has a degree of fulfillment higher than a certain threshold ζ (i.e. only on those $\mathbf{x} \in \mathcal{X}$ for which $n^j(\mathbf{x}) > \zeta$). Typically, we choose $\zeta = 0.1$.

4.4 Description of the optimization algorithm – *LAPOC*

The algorithm for optimizing TSK systems described above consists of two main parts. First, the system of the optimization problem is set up, then this system is solved. We will now present a rough schedule of the algorithm ultimately obtained, *LAPOC* (**L**inear **Ap**proximation **o**f **C**onsequences).

```
Algorithm 3 (LAPOC)
  Input:     samples $\mathcal{X} = \{\mathbf{x}^1, \ldots, \mathbf{x}^K\}$
             rule base $R$
  Output:    optimized rule base $\hat{R}$

  Generate the system matrix $N^{obs}$ as in (15)
  Set observation vector $\mathbf{y} = \mathbf{x}_{n+1}$

  if variable selection is desired
  {
      for each rule
          Apply Forward Selection
              to identify the variables to use
          Delete entries of $N^{obs}$ corresponding to
              variables which were not chosen
  }

  Apply Tikhonov-regularization to the achieved system
      Result:   $N^{reg}$ according to (16)
  Solve the linear least squares problem according to (18)
      $\boldsymbol{\alpha} = \mathbf{LinearSolve}(N^{reg}, \mathbf{y})$
  Modify input rule base
      $\hat{R} = R(\boldsymbol{\alpha})$
```

The solving algorithm we use reduces the observation matrix to an upper triangular form by a series of Givens rotations and is adapted to sparse matrices. When applying variable selection, in general we do not use the whole data set for selecting the most decisive variables, but we choose only a reduced data set. For each rule, we use only those samples for which the corresponding rule output is at least of a certain "minimum degree of membership". This improves the performance of the algorithm without any loss of accuracy.

For the local approach, the algorithm is almost the same, except that we perform the algorithm individually for each rule.

5 Experimental results

In this section, we will present some experiments which give an overview over special characteristics of our methods and illustrate the performance of the proposed methods. For this purpose, we used four different data sets. First, we used the *housing data set* from the UCI repository (Blake and Merz 1998). Then we defined two different *two-dimensional* test functions:

$$f_{2D-A}(\mathbf{x}) = x_2 \cdot \sin(2\pi x_1), \qquad \mathbf{x} \in [0,1]^2$$
$$f_{2D-B}(\mathbf{x}) = \frac{\cos\left(5\pi(x_1^2+x_2^2)^{1/2}\right)}{(1+10(x_1^2+x_2^2)^{1/2})}, \qquad \mathbf{x} \in [-1,1]^2,$$

Finally, we defined a *six-dimensional* test function

$$f_{6D}(\mathbf{x}) = x_1 + x_2^{0.5} + x_3 x_4 + 2e^{2(x_5 - x_6)},$$

with $\mathbf{x} \in [1,5] \times [1,5] \times [0,4] \times [0,0.6] \times [0,1] \times [0,1.2]$. To validate the models, we used 10-fold cross validation to determine the average model complexity (in terms of number of leaves) and the average model performance (correlation coefficient between the actual output and the predicted output on the test data).

5.1 On the choice of fuzzy sets

To answer the question of how to choose the number and shape of fuzzy sets for a given problem, we ran several experiments with different settings on the four data sets using *FS-LiRT* with constant output values and without pruning.

To illustrate the expressive power of the ordering-based predicates, we performed a series of analyses where we used only the predicates directly induced by the corresponding fuzzy sets first and performed the analyses again by using all the predicates according Eq. (8) afterwards. Results are shown in Fig. 3 and Fig. 4, respectively. It shows that by using ordering-based predicates, the complexity of the resulting model and the

Fig. 3 Influence of the number of input sets on the correlation of the resulting model using IS predicates only

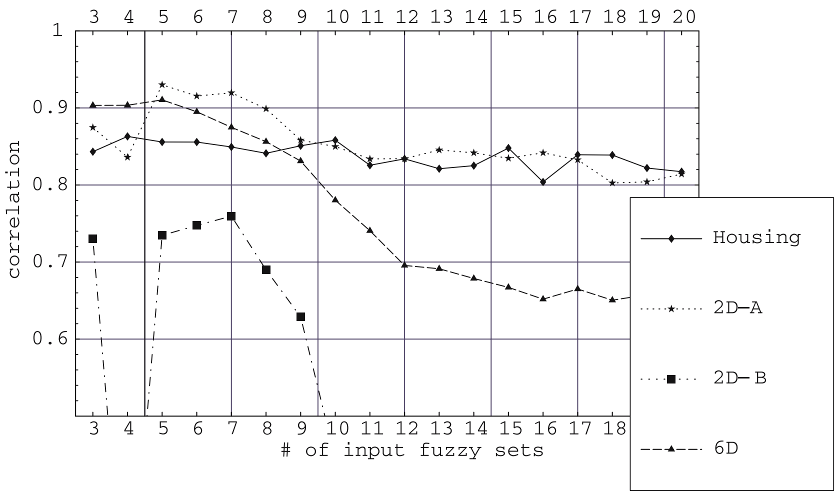
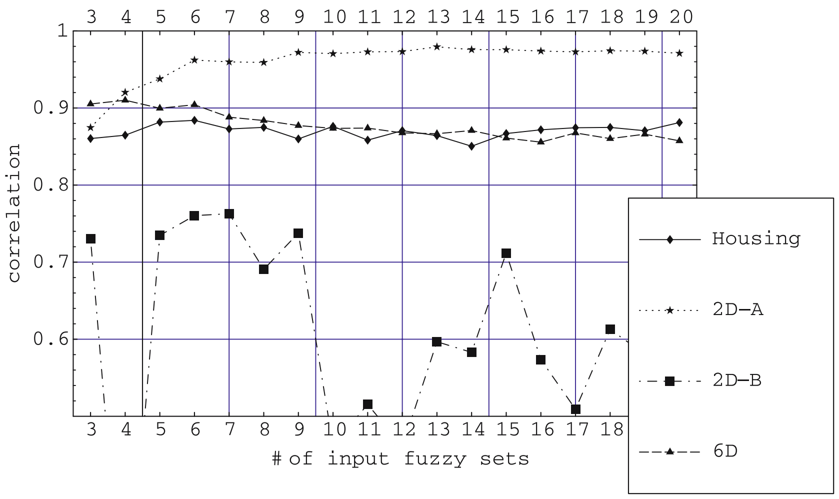


Fig. 4 Influence of the number of input sets on the correlation of the resulting model using ordering-based predicates



quality of the results obtained are much less influenced by the number and structure of the underlying fuzzy sets as compared with using only simple predicates. Only the results for function f_{2D-B} are influenced by the number of sets as this function is highly non-linear with large local deviations. Therefore, the choice of the underlying fuzzy sets is crucial to describing this behavior. The overall performance is, however, still more robust than without ordering-based predicates.

When increasing the number of fuzzy sets for the input attributes, one would expect that the resulting models get more complex. It turns out, however, that by using ordering-based predicates, the complexity of the models computed remains almost constant. A comparison of experiments using different data sets is shown in Fig. 5. This can be explained by the fact that using ordering-based predicates, we obtain more predicates with a high support, even if the number of sets is high. Furthermore, the support of the rules composed only of simple predicates decreases when the number of sets increases. Therefore, more rules have to be created to obtain a complete coverage of the state space, or, if these rules do not fulfill the minimum requirements in terms of support, the algorithm fails.

Finally, we investigated the influence of the shape of the underlying fuzzy sets on both the accuracy and the complexity of the created models (Fig. 6). In our tests, we compared a crisp partitioning ($o = 0$) with fuzzy partitions using fuzzy sets with a piece-wise linear membership function ($o = 0.2$, $o = 0.5$, $o = 0.8$, and $o = 1.0$) and fuzzy sets with an exponential membership function. It turns out that fuzzy sets with a high overlap are advantageous, especially for highly non-linear problems. The results are shown in Fig. 6.

5.2 Examples for the influence of regularization

We will now have a closer look on the aspect of numerical stability; i.e. we show that applying regularization

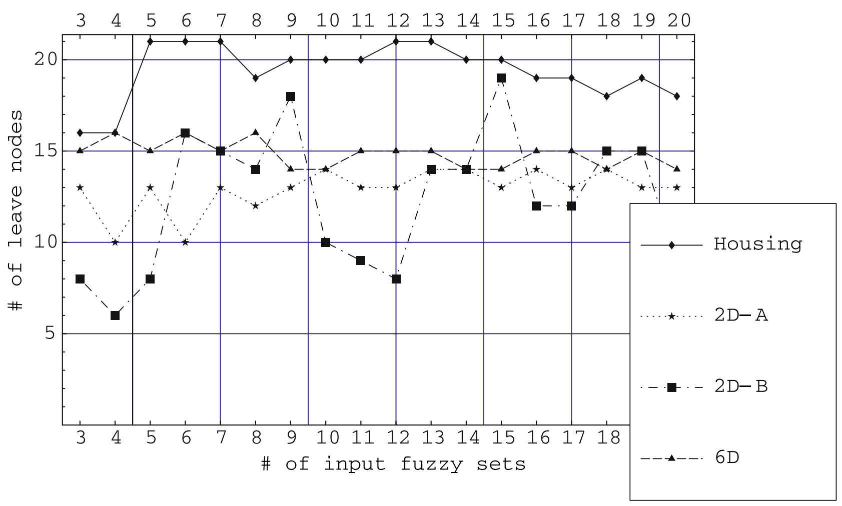


Fig. 5 Influence of the number of input sets on the complexity of the resulting model using ordering-based predicates

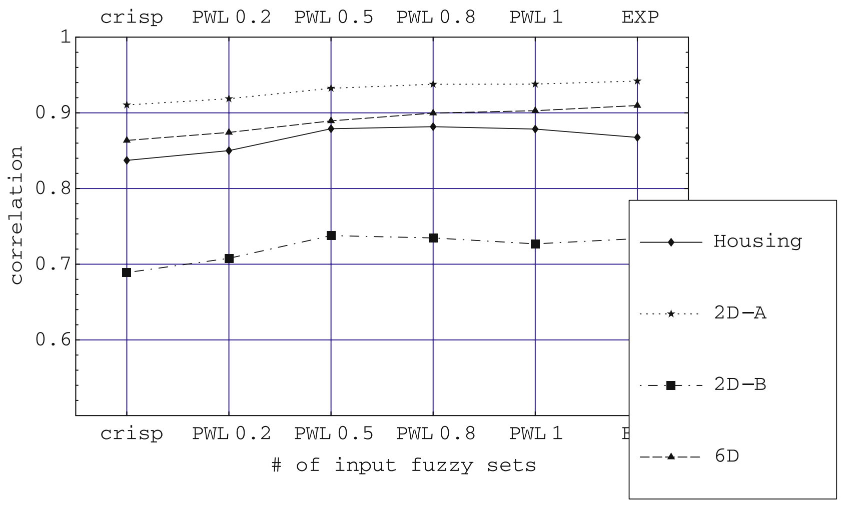


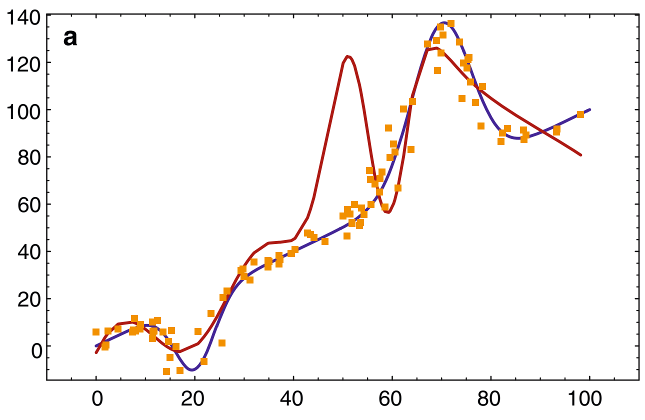
Fig. 6 Influence of the shape of the input sets on the correlation of the resulting model using five fuzzy sets for all input attributes

to our optimization problems was essential in order to obtain stable solutions. Note, however, that the choice of the regression parameter for the optimization process is crucial, as it strongly influences the performance of the algorithm.

As already mentioned, in the case of Tikhonov regularization, the original problem is modified by adding a penalty term the Euclidean norm of the vector of consequences. For $\beta \rightarrow \infty$ and $\alpha_k \rightarrow 0$ for all k, as well as for (appropriately chosen) $\beta \rightarrow 0$, the solution converges to the one of the unregularized problems. For sufficiently high β, we can expect a stable and smooth solution, however less accurate. Decreasing the value of β results in increasing accuracy, however at the expense of the system's stability. We will see that in many cases, the choice of the regularization parameter is of crucial importance for the solution process.

Figure 7 demonstrates how regularization influences the result in the case of noisy data. LAPOC was applied to a noisy data set (dots), generated by an one-dimensional function with two major peaks, with several values of β. The left side shows the approximation achieved with $\beta = 0$ (no regularization), whereas on the right side the best result ($\beta = 0.001$) is shown. Without regularization, an artificial third peak appears, i.e. the solution becomes unstable.

Figure 8 shows the correlation coefficient between the original goal attribute and the model output for decreasing β for a test run of LAPOC applied to the housing data set. The solid lines are the values for the training data, the dashed lines for the test data. The lighter lines show the results of LAPOC with full approximation, the darker lines the results obtained by LAPOC with variable selection. For $\beta < 0.1$, the test results of LAPOC deteriorate rapidly, both as a cause of overfitting and increasing system condition number. Moreover, when setting β to zero (i.e. no regularization applied), the correlation coefficient on the training data decreases, too.



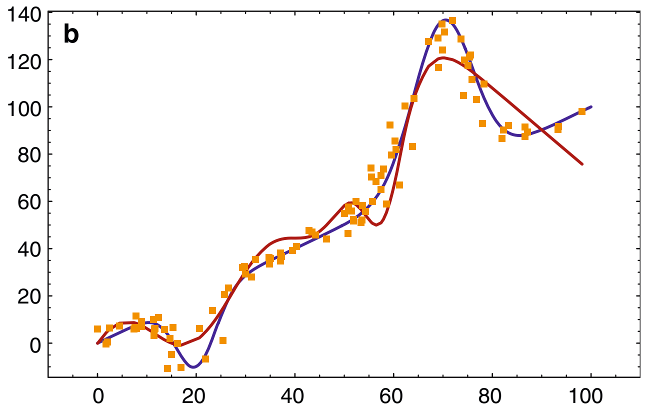


Fig. 7 Approximation of a one-dimensional function with noisy training data using no regularization (*left*) and with regularization (*right*)

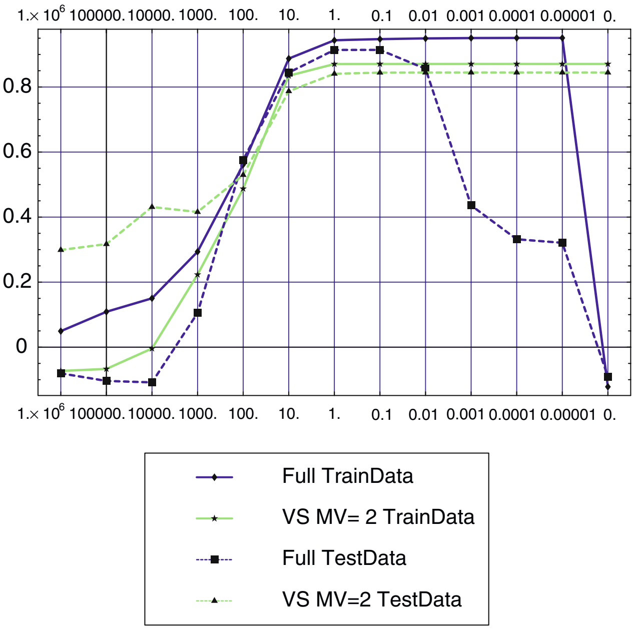


Fig. 8 Correlation coefficients on training and test data for different values of the regularization parameter β for the *housing data set*

A closer look on the rank of the system matrix shows that this was because we ended up with a rank-deficient matrix N^{obs} $(89 = r(N^{\text{obs}}) < l_L(n+1) = 98)$.

5.3 Examples

In this section we will present experiments with *LAPOC* and *FS-LiRT* applied to the *housing data set* and to the artificial *six-dimensional* function. A comparison of the results shows that with the same requirements to accuracy we can create smaller, highly interpretable and more expressive models with *LAPOC*.

Example 1 (UCI-Data (Housing)) As already mentioned before, in this example we investigated how much we can simplify a regression tree without reducing its accuracy using global optimization. Using *FS-LiRT* with constant output values for the computation of a regression tree for the housing data set, we achieved an average correlation coefficient of 0.859 at an average model size of 11 (Fig. 9). Although, due to the use of fuzzy sets, the information of one record might be spread over the whole tree, a large number of records will belong to one leaf node with a very high degree of membership. As in many cases ordering based predicates (which have a large kernel) are used, the regions where the individual branches overlap do not influence the model structure dramatically. They have, however, a large influence on the application of the model, as these overlaps enable us to create smooth transitions from one state to another.

We achieved the same accuracy (0.858) by optimizing a regression tree of average model size 4 using linear approximation of the output functions with at most four variables (of overall 13 input dimensions) for each rule. The extracted rule base of a partial result is shown in Fig. 10. Although the rule outputs might look a bit complicated, the terms in brackets can be replaced with according variables to simplify the output (e.g. replacing $(B - 306.775)$ in the first rule's consequence with B'). These terms, however, contain some important information, too, as e.g. the term $(B - 306.775)$ tells us that the attribute B has an average value of 306.775 over all samples fulfilling the first rule's antecedent. Furthermore it is, at least for a rough overview, sufficient to consider the constant rule outputs, as the linear terms are only additional information which does not alter the core information of the rule. In Fig. 11 the result obtained using *M5P-Rules* is shown. It is easy to see, that these rules are far more complex then those obtained using *FS-LiRT*.

Example 2 *(Approximation of the six-dimensonal function.)* Figure 12 shows a comparison of experiments using models of different sizes for the basic

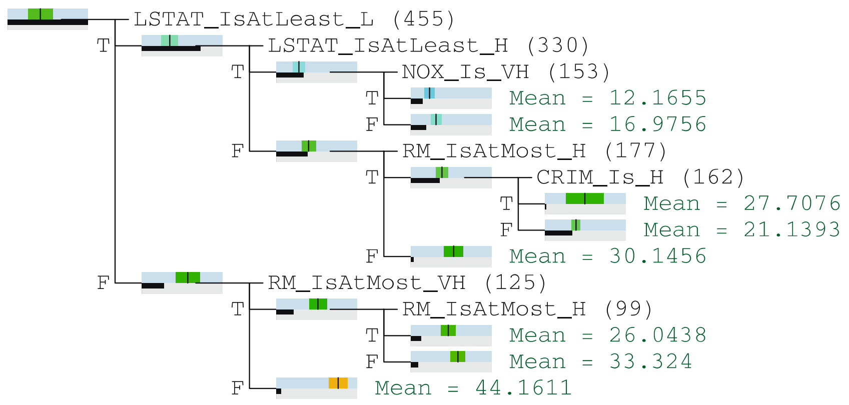


Fig. 9 Regression tree generated by FS-LiRT for the housing data set

Fig. 10 LAPOC applied to a rule base extracted from a tree generated by *FS-LiRT* for the housing data set

RM_IsAtMost _H && LSTAT_IsAtLeast _H	⟹	14.2517 + 0.00422881 (B - 306.775) + -0.173667 (CRIM - 8.80581) + 0.926575 (DIS - 2.48377) + -0.605399 (RM - 5.87854)
RM_IsAtMost _H && ~ [LSTAT_IsAtLeast _H]	⟹	23.1316 + 4.48427 (CHAS - 1.06669) + -1.01355 (LSTAT - 9.44672) + 0.214037 (RAD - 7.13699) + 0.580859 (RM - 6.24525)
~ [RM_IsAtMost _H] && PTRATIO_IsAtMost _M	⟹	41.1863 + -0.249751 (RM - 7.59403)
~ [RM_IsAtMost _H] && ~ [PTRATIO_IsAtMost _M]	⟹	15.5342 + 0.204839 (CRIM - 8.65116) + -12.0671 (DIS - 2.48421) + 1.83365 (RAD - 21.8046) + -2.14279 (RM - 7.6637)

regression tree. The graphic also illustrates how optimization improved the accuracy of the basic model (created by *FS-LiRT* with constant output values). Note that with increasing model size the improvement of the correlation coefficient decreased. The same correlation coefficient, however, was never reached with basic *FS-LiRT*. The global and the local optimization (*LAPOC* and *LiRT Ml*, respectively) had nearly the same performance for all experiments. In this example, the use of variable selection always slightly impaired the performance of the optimization, but, of course, for the benefit of simpler output functions.

5.4 Comparison with other algorithms

To compare the performance of the proposed methods with other approaches, we used eleven regression problems from the UCI repository (see Table 1). Again, we used ten fold cross validation and computed the average correlation coefficient σ. We compared the results

Table 1 Data sets used

Data set	No. Recs.	No. Attr.	Num. Attr.	Cat. Attr.
2D-A	100	3	3	0
2D-B	1,000	3	3	0
6D	1,000	7	7	0
AutoPrice	159	16	16	0
Bodyfat	252	15	15	0
Bolts	40	8	8	0
Cloud	108	7	5	2
Cpu	209	8	7	1
Detroit	13	14	14	0
Gascons	27	5	5	0
Housing	506	14	14	0
Longley	16	7	7	0
Pollution	60	16	16	0
PwLinear	200	11	11	0

a

```
LSTAT > 9.725     =>  -0.1241 * CRIM
LSTAT ≤ 15            + 0.0022 * ZN
                      - 2.1936 * NOX
                      + 1.8229 * RM
                      - 0.0319 * AGE
                      - 0.7919 * DIS
                      + 0.0355 * RAD
                      - 0.0016 * TAX
                      - 0.4903 * PTRATIO
                      + 0.001 * B
                      - 0.0619 * LSTAT
                      + 26.2232
------------------------------------------------
LSTAT > 12.365    =>  -0.2769 * CRIM
CRIM ≤ 5.769          + 0.0032 * ZN
CRIM > 0.654          - 0.0247 * INDUS
                      - 0.7684 * NOX
                      + 0.3339 * RM
                      - 0.0262 * AGE
                      - 0.1386 * DIS
                      + 0.1463 * RAD
                      - 0.0011 * TAX
                      - 0.0971 * PTRATIO
                      + 0.0042 * B
                      - 0.1146 * LSTAT
                      + 19.8551
------------------------------------------------
LSTAT > 12.365    =>  -0.0292 * CRIM
TAX ≤ 551.5           + 0.0037 * ZN
                      - 17.7675 * NOX
                      + 0.4319 * RM
                      - 0.1761 * DIS
                      + 0.0613 * RAD
                      - 0.0043 * TAX
                      - 0.1218 * PTRATIO
                      + 0.0011 * B
                      - 0.2551 * LSTAT
                      + 34.4628
------------------------------------------------
LSTAT > 12.365    =>  -0.0648 * CRIM
LSTAT > 19.73         + 0.0062 * ZN
NOX > 0.675           - 14.9901 * NOX
                      - 0.3298 * RM
                      + 0.0229 * AGE
                      + 1.4827 * DIS
                      + 0.0602 * RAD
                      - 0.0013 * TAX
                      - 0.1747 * PTRATIO
                      - 0.0003 * B
                      - 0.3182 * LSTAT
                      + 29.7017
```

b

```
RM ≤ 6.941        =>  3.0866 * CRIM
LSTAT ≤ 12.355        + 0.0053 * ZN
DIS > 3.325           - 0.0173 * INDUS
RM ≤ 6.545            - 23.5392 * NOX
                      + 5.0937 * RM
                      - 0.0219 * AGE
                      - 0.628 * DIS
                      + 0.1203 * RAD
                      - 0.0136 * TAX
                      - 0.2046 * PTRATIO
                      + 0.0017 * B
                      - 0.2992 * LSTAT
                      + 14.9422
------------------------------------------------
LSTAT ≤ 12.365    =>  0.9478 * CRIM
RM ≤ 7.437            - 0.0646 * INDUS
DIS > 2.701           - 5.1334 * NOX
                      + 7.4386 * RM
                        0.0389 * AGE
                      - 0.6318 * DIS
                      + 0.0494 * RAD
                      - 0.0133 * TAX
                      - 0.2722 * PTRATIO
                      - 0.0109 * B
                      - 0.7119 * LSTAT
                      + 3.5661
------------------------------------------------
LSTAT > 9.63      =>  -10.7764 * NOX
                      + 1.1535 * RM
                      + 0.0051 * TAX
                      - 0.4787 * PTRATIO
                      + 0.0029 * B
                      - 0.2273 * LSTAT
                      + 24.7003
------------------------------------------------
LSTAT > 5.205     =>  3.5849 * CRIM
TAX > 268.5           + 2.1721 * RM
                      - 0.0212 * AGE
                      - 0.0254 * TAX
                      - 0.4544 * PTRATIO
                      - 0.9528 * LSTAT
                      + 35.1797
------------------------------------------------
Default           =>  2.5058 * CRIM
                      + 3.6052 * RM
                      - 0.0685 * AGE
                      - 0.4884 * PTRATIO
                      - 0.144 * B
                      - 0.8388 * LSTAT
                      + 86.699
```

Fig. 11 Regression model generated by M5P-rules for the *housing data set*

with three methods from the Weka toolkit (Witten and Frank 2000), namely *Linear Regression*, *M5-Prime* [a variant of *C4.5* (Quinlan 1992)], 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 to do so. We ran *M5-Prime* using constant output values and linear models, with smoothing enabled. We used the Weka Toolbox 3–4 to obtain the final results.

First, we compared the average correlation coefficient between the original output and the predicted values for the test data. The results are shown in Table 2. We can see that for these data sets, our methods performed equally well as the other methods. Comparing *FS-LiRT* with constant and with linear output models shows that

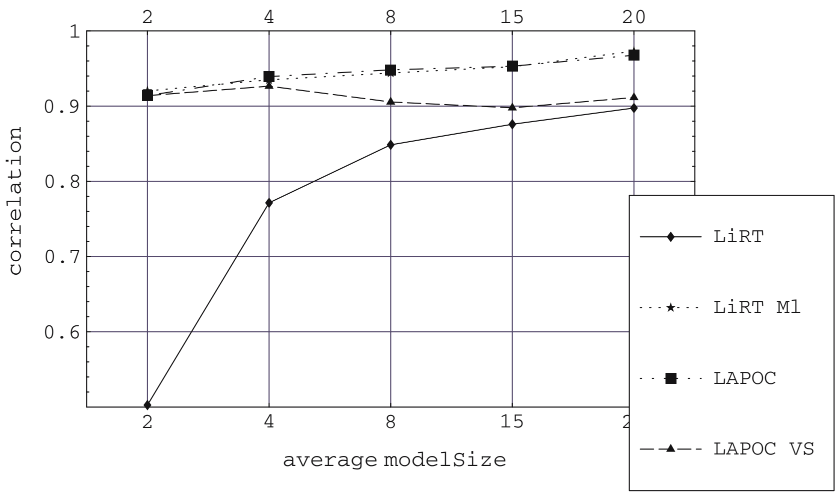


Fig. 12 Correlation for models with different complexities for the six-dimensonal test function

Table 2 Average correlation coefficients achieved using ten fold cross validation

Method/Data Set	LiRT	LiRT Model	M5P Model	M5P	M5Rules	Lin.Reg.
2D-A	0.93	0.97	0.99	0.98	0.97	0.65
2D-B	0.84	0.88	0.91	0.88	0.89	–
6D	0.93	0.99	0.99	0.92	0.9	0.91
AutoPrice	0.87	0.91	0.93	0.86	0.92	0.87
Bodyfat	0.92	0.93	0.99	0.96	0.99	0.99
Bolts	0.91	0.98	0.91	0.87	0.9	0.93
Cloud	0.82	0.88	0.92	0.72	0.87	0.92
Cpu	0.83	0.95	0.97	0.78	0.97	0.93
Detroit	0.76	0.76	0.84	–0.29	0.84	0.67
Gascons	0.96	0.96	0.86	0.87	0.93	0.56
Housing	0.86	0.88	0.91	0.87	0.9	0.85
Longley	0.94	0.98	0.97	0.72	0.97	0.98
Pollution	0.56	0.51	0.71	0.5	0.71	0.74
PwLinear	0.85	0.94	0.94	0.84	0.94	0.85

the latter performs significantly better (two-sided t test on the mean of differences $H_0 : d(m_1, m_2) = 0$, $p = 0.024$). When comparing *FS-LiRT* with constant output functions with *M5-Prime* with constant output functions, *FS-LiRT* performs slightly better ($p = 0.156$). Compared to *M5-Prime* with linear output models, *FS-LiRT* has almost the same performance ($p = 0.32$). We can, however, observe that the results obtained using simple linear regression have the same performance, too. This indicates that these problems are almost linear and complex methods do not provide any additional benefit at all. For the complex, two-dimensional problems where oscillations are involved, the more complex methods outperform linear regression easily. In these cases, our methods perform also much better then *M5-Prime* and *M5-Rules*. This is mainly caused by the fact that using exponential type fuzzy sets enables us to create smooth transitions between the different rules.

Finally, we compared the corresponding sizes of the models learned. The results are shown in Table 3. Again, we can see that for the UCI data sets, the size of the resulting models is almost equal but for the artificial data sets, *FS-LiRT* was able to extract much more compact representations. As the models created by our methods are, however, based on a predefined, well-structured language, they are much easier to comprehend. Furthermore, we ran all experiments using the default settings of the algorithms. By adjusting these settings, further improvements can be achieved.

6 Conclusions and further work

In this work, we have presented a novel approach to derive comprehensible *and* numerically accurate fuzzy regression models from data. Comprehensibility is achieved by using a novel method to derive fuzzy sets from a given data set and by introducing ordering-based predicates. It was shown that the actual choice of the number and shape of the fuzzy sets used has only a minor influence on the performance of the resulting model. To

Table 3 Average model sizes achieved using 10 fold cross validation

Method / Data Set	LiRT	LiRT Model	M5P Model	M5P	M5Rules
2D-A	9	6	153	102	86
2D-B	69	52	100	112	29
6D	56	18	150	18	51
AutoPrice	4	8	7	9	4
Bodyfat	3	3	4	17	5
Bolts	4	3	3	3	3
Cloud	5	4	3	8	3
Cpu	2	2	3	13	3
detroit	6	7	1	2	1
Gascons	3	3	3	3	2
Housing	6	9	13	21	8
Longley	4	2	1	3	1
Pollution	12	12	1	4	1
PwLinear	7	4	2	12	2

inductively learn regression models from data, we have presented a new method which uses fuzzy decision trees with linear models in the leaf nodes. Variable selection is applied to increase interpretability of these linear terms. High accuracy of these models is achieved by applying regularized numerical optimization techniques, either in the leaf nodes or on the complete model. It has turned out that using regularization techniques, global optimization leads to almost equal results as local optimization. The choice of the regularization parameter is, however, crucial and depends heavily on the data under investigation.

The presented approach has been applied on various problems in all kind of application areas, ranging from metallurgy and paper industry to energy production. In all these applications, we have received very positive feedback for the comprehensibility of the models and the achieved accuracy.

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, 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 The results presented in this paper have been computed using the *machine learning framework for Mathematica (mlf)*. A trial version of *mlf* can be downloaded at `http://www.unisoftwareplus.com/products/mlf`. *mlf* is developed and supported by the knowledge based technology area of SCCH[1]. *mlf* is owned and distributed by *uni software plus GmbH*[2], which collaborates with *Wolfram Research, Inc.*, for worldwide distribution.

Mario Drobics and Johannes Himmelbauer gratefully acknowledge support by the Austrian Government, the State of Upper Austria, and the Johannes Kepler University Linz in the framework of the K*plus* Competence Center Program.

References

Adamo JM (1980) Fuzzy decision trees. Fuzzy sets Sys, 4:207–219

Baldwin JF, Lawry J, Martin TP (1997) A mass assignment based ID3 algorithm for decision tree induction. Int J Intell Syst 12:523–552

Baranyi P, Yam Y, Tikk D, Patton RJ (2003) Trade-off between approximation accuracy and complexity: TS controller design via HOSVD based complexity minimization. In: Casillas J, Cordón O, Herrera F, Magdalena L (eds), Interpretability Issues in Fuzzy Modeling, of Studies in Fuzziness and Soft Computing, Springer, Berlin Heidel berg New York Vol 128, pp, 249–277

Blake CL,Merz CJ (1998)UCI repository of machine learning databases. University. of California, Irvine, Dept of Information and Computer Sciences http://www.ics.uci.edu/∼mlearn/MLRepository.html

Bodenhofer U (1999a) The construction of ordering-based modifiers In Brewka G, Der R, Gottwald S, Schierwagen A, (eds) Fuzzy-Neuro Systems '99, Leipziger Universitätsverlag pp 55–62

Bodenhofer U (199b) A Similarity-Based Generalization of Fuzzy Orderings, vol C 26 of Schriftenreihe der Johannes-Kepler-Universität Linz. Universitätsverlag Rudolf Trauner

Bodenhofer U, Bauer P (2003) A formal model of interpretability of linguistic variables. In: Casillas J, Cordón O, Herrera F, Magdalena L (eds) Interpretability issues in fuzzy modeling. vol 128 of Studies in fuzziness and soft computing, Springer, Berlin Heidel berg New York pp 524–545

[1] Software Competence Center Hagenberg GmbH (A-4232 Hagenberg, Austria, `http://www.scch.at/kbt`).

[2] uni software plus GmbH (Kreuzstrasse 15a, A-4040 Linz, Austria, `http://www.unisoftwareplus.com`).

Breiman L, Friedman J, Stone CJ, Olshen RA, (eds)(1984) Classification and regression trees. CRC Press Bow, Raton

Burger M, Haslinger J, Bodenhofer U, Engl HW (2002) Regularized data-driven construction of fuzzy controllers. J Inverse Ill-Posed Probl, 10(4):319–344

Casillas J, Cordón O, Herrera F, Magdalena L (eds) (2003) Interpretability issues in fuzzy modeling, vol 128 of Studies in fuzziness and soft computing. Springer, Berlin Heidelberg Newyork

Castellano G, Fanelli AM, Mencar C (2002) A double-clustering approach for interpretable granulation of data. In: Proceeding of 2002 IEEE International Conference on systems, man and cybernetics, Hammamet, Tunisia

de Boor C (1998) A practical guide to splines. Springer, Heidelberg New York

De Cock M, Bodenhofer U, Kerre EE (2006) Modelling linguistic expressions using fuzzy relations. In: Proceedings of 6th International. Conference. on Soft Computing, pp 353–360, Iizuka

Draper NR, Smith H, Applied Regression Analysis. John Wiley & Sons, New York

Drobics M (2004) Choosing the best predicates for data-driven fuzzy modeling. In: Proceedings of . 13th IEEE international conference. on fuzzy systems, Budapests pp 245–249

Guillaume S, Charnomordic B (2004) Generating an interpretable family of fuzzy partitions from data. IEEE Trans Fuzzy Syst, 12(3)

Höppner F, Klawonn F (2005) Improved fuzzy partitions for fuzzy regression models. Internat J Approx Reason 32:85–102

Janikow CZ, Fuzzy decision trees: issues and methods. IEEE Trans Syst Man Cybern B, 28(1):1–14

Klawonn F, Kruse R (1997) Constructing a fuzzy controller from data. Fuzzy Sets Systems, 85:177–193

Maher PE, Clair DS, (1993) Uncertain reasoning in an ID3 machine learning framework. In: Proceeding of 2nd IEEE international Conference on fuzzy systems, San Francisco, CA

Marsala C (2000) Fuzzy decision trees to help flexible querying. Kybernetika, 36(6):689–705

McClelland JL, Rumelhart DE,(eds) (1996) Parallel distributed processing—exploration in the Microstructures of Cognition, Vol II: Psychological and Biological Models. MIT Press, Cambridge,

Mikut R, Jäkel J, Gröll L (2000) Automatic design of interpretable membership functions. In: Proceeding of 8th zittau fuzzy colloquium, Hochschule Zittau/Görlitz

Mikut R, Jäkel J, Gröll L, Interpretability issues in data-based learning of fuzzy systems. Fuzzy Sets and Syst, 150:179–197

Nelles O, Fink A, Isermann R (2000) Local linear model trees (LOLIMOT) toolbox for nonlinear system identification. In: Proceeding of 12th IFAC symposium on system Identification, Santa Barbara

Olaru C, Wehenkel L (2003) A complete fuzzy decision tree technique. Fuzzy Sets Syst 138(2):221–254

Peng Y, Flach PA (2001) Soft discretization to enhance the continuous decision tree induction. In: Proceeding of ECML/PKDD01 workshop integrating aspects of data mining, decision support and meta-learning, PP 109–118

Quinlan JR (1992) Learning with continuous classes. In: Proceeding of 5th Australin Joint Conference on artificial intelligence, pp 343–348

Quinlan JR, C4.5: Programs for machine learning. Morgan Kaufmann, San Mateo

Regattieri Delgado JR, Von Zuben F, Gomide F (2001) Local and global estimation of Takagi-Sugeno consequent parameters in genetic fuzzy systems. In: Proceeding of Joint 9th IFSA world congress and 20th NAFIPS International Confference, Vancouver

Rumelhart DE, McClelland JL, Parallel distributed processing—exploration in the microstructures of cognition, Vol I: Foundations. MIT Press, Cambridge

Takagi T, Sugeno M (1985) Fuzzy identification of systems and its applications to modeling and control. IEEE Trans Syst Man Cybern 15(1):116–132

Wang X, Chen B, Qian G, Ye F (2006) On the optimization of fuzzy decision trees. Fuzzy Sets Syst 112:117–125

Witten IH, Frank E Data mining: *practical machine learning tools with Java implementations*. Morgan Kaufmann

Yager RR, Filev E, Sanmateo (1994) Approximate clustering via the mountain method. IEEE Trans Syst Man Cybern 24(8):1279–1284

Yuan Y, Shaw MJ (1999) Induction of fuzzy decision trees. Fuzzy Sets and Syst, 69:125–139

Zadeh LA, Fuzzy sets. Inf. Control 8:338–353

Zeidler J, Schlosser M (1996) Continuous valued attributes in fuzzy decision trees. In: Proc 8th Int conf. on information processing and management of uncertainty in knowledge-based systems, pp 395–400

Zurada JM, Introduction to Artificial Neural Networks. West Publishing, St. Paul