Identifying Rule-Based TSK Fuzzy Models

Manfred Männle
Institute for Computer Design and Fault Tolerance
University of Karlsruhe
D-76128 Karlsruhe, Germany
Phone: +49 721 608-6323, Fax: +49 721 608-3962
Email: maennle@computer.org

ABSTRACT: This article presents a rule-based fuzzy model for the identification of nonlinear MISO (multiple input, single output) systems. The presented method of fuzzy modeling consists of two parts: (1) structure modeling, i.e., the determination of the number of rules and input variables involved respectively, and (2) parameter optimization, i.e., the optimization of the location and form of the curves which describe the fuzzy sets and the optimization of the consequence parameters. For structure modeling we use a modified TSK-model, which was first proposed by Takagi, Sugeno, and Kang in [Takagi 85, Sugeno 88]. For parameter optimization of the initial model we propose singular value decomposition (SVD). To optimize the parameters of further models we propose the use of RPROP [Riedmiller 93, Zell 94]. We applied RPROP to the modified version of the TSK-model, implemented the algorithm, and tested its performance [Männle 95, Männle 96]. In this article we focus on the structure modeling part. By means of two examples we show the performance of RPROP and the input space partitioning performed by the structuring algorithm.

KEYWORDS: Nonlinear system identification, rule-based fuzzy models, input space partitioning, learning algorithms

1 INTRODUCTION

Fuzzy theory finds its application to deal with problems where there is imprecision caused by the absence of sharp criteria [Zadeh 65]. An example for this is human language processing. Linguistic terms can be defined by fuzzy sets and we can formulize fuzzy if-then rules [Zadeh 73]. Operators like AND and OR and the implication IF ... THEN are defined and so we can somehow *calculate* with statements given in this form.

One important practical application of fuzzy theory is *fuzzy control*: The desired control actions are described by human experts in the form of fuzzy if-then rules. Combined with fuzzification, a fuzzy inference machine, and defuzzification, a set of fuzzy if-then rules provides a fuzzy controller, which showed itself as a robust and powerful controller in many applications.

In this paper we describe a modeling approach. We exploit measured input-output data of an unknown process in order to get an interpretable process description. Therefore, we automatically generate fuzzy if-then rules which describe the process' input-output behavior. This is commonly known as *fuzzy modeling* or fuzzy identification [Zadeh 94].

In reality, the model response is an approximation to the system response which holds for a restricted set of inputs. Such an approximation can be designed at an arbitrary precision (e.g., using a polynomial covering all data points), but usually you then have identified the set of inputs and not the underlying process. What we are searching here is an *interpretable model*, i.e., a qualitative description of the underlying process. For this task fuzzy modeling has been proven to be a suitable approach.

There are various kinds of fuzzy modeling. One is to describe the input-output relation of data with a fuzzy relation. *Relation-based* approaches are for example described in [Pedrycz 91, Vrba 92, Chen 94]. Some comparisons of relation-based approches can be found in [Postlethwaite 91, Böhler 94].

Another approach is to divide the input-output space into *clusters*. Fuzzy rules are generated by projection of these clusters to the input space universes. Each rule represents one or several clusters which can be interpreted as local models. For clustering, Sugeno and Yasukawa use the fuzzy c-means method [Sugeno 93], Nakamori and Ryoke use fuzzy c-varieties [Nakamori 94].

In this work we have to deal with another problem. We want to identify complex systems with a *a large number of input variables*. For this, methods yielding a polynomial or even exponential number of rules (depending from the number of input variables) like relation-based approaches are not applicable. That is why we take a "bottom up" rule-based approach: the algorithm starts with a one-rule model and adds additional rules at each refining step until the desired accuracy or a maximum number of rules is reached. The model structure we use here is similar to the TSK-model's [Zadeh 94], which

was first proposed by Takagi, Sugeno, and Kang in [Takagi 85, Sugeno 88]. Other derivations from TSK can be found for example in [Yager 93, Tanaka 94, Jang 93, Yen 98].

In Section 2 we describe the modified TSK-model we use for fuzzy modeling. We also explain in detail the algorithm's search for the model structure and the parameter optimization of the initial model by singular value decomposition (SVD). For a description of RPROP, the parameter optimization for the defined models, we refer to [Männle 95, Männle 96]. In Section 3 we demonstrate the way RPROP works and how the structuring algorithm performs an input space partitioning. As a first example we choose a step function in order to show the performance of the RPROP parameter optimization. This type of function is relatively hard to approximate for a model having smooth sigmoidal membership functions, since the optimization technique has to find a very high steepness of the membership function. The second example demonstrates how the algorithm and especially the input space partitioning works. For this purpose we choose two Gaussian bell functions as a more complex example of a two dimensional mapping. Section 4 closes this article by some conclusions and perspectives.

2 FUZZY IDENTIFICATION

The modeling is performed iteratively. Each iteration consists of two phases (see figure 1): Beginning from a simple model, a new and more complex structure (containing more fuzzy rules) is chosen and its parameters are optimized by a learning algorithm based on the given training data. This step is repeated until the model is considered good enough or the border for a maximum number of fuzzy rules is exceeded.

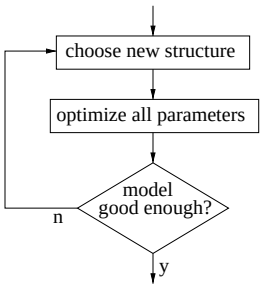

Figure 1: Basic scheme of the modeling procedure.

The training data consists of M measured or simulated input-output data pairs $(\vec{u}_1, y_1), \ldots, (\vec{u}_M, y_M)$ with $\vec{u}_m \in I\!R^N$, $y_m \in I\!R$. For $m = 1, \ldots, M$, the fuzzy model performs a nonlinear mapping

$$\hat{y}_m = \hat{f}(\vec{u}_m) \stackrel{!}{\approx} y_m, \tag{1}$$

i.e., it models a MISO system with N input variables. MIMO systems can be identified by applying several MISO models.

2.1 The Fuzzy Model

The fuzzy model used is derived from the fuzzy model of Takagi, Sugeno, and Kang [Takagi 85, Sugeno 88], which is often referred to as TSK-model. The whole fuzzy model consists of a set of R fuzzy rules $R_1, \ldots, R_R$. Each rule consists of a premise and a consequence. Consequences of Sugeno-like fuzzy models like the TSK model are simply a linear combination of (possibly but not necessarily) all input variables. Therefore, each consequence defines a linear mapping or, geometrically speaking, a hyperplane in the input-output space.

The premise of the fuzzy rule $R_r,\ 1 \leq r \leq R$ is a conjunction of n_r fuzzy clauses of the form $u_{i_j r}$ `is` F_{jr}, i.e., it considers those n_r input variables, which are given in the index set I_r. Using an index set for each rule is necessary, since a fuzzy rule needs not to be complete, i.e., needs not to consider all input variables $u_j, j = 1, \ldots, N$. As we will see later, each premise defines a subspace of the input space. Therefore, Sugeno-like models can be seen as piecewise linear models with fuzzy transitions.

The initial fuzzy rule R_1 has for $I_1 = \emptyset$ the form

$$\texttt{if TRUE then } f_1 = p_{01} + p_{11} \cdot u_1 + \ldots + p_{N1} \cdot u_N,$$

and has for $I_r \neq \emptyset$ the form

$$\texttt{if } u_{i_{1r}} \texttt{ is } F_{1r} \texttt{ and} \ldots \texttt{and } u_{i_{n_r r}} \texttt{ is } F_{n_r r} \texttt{ then } f_r = p_{0r} + p_{1r} \cdot u_1 + \ldots + p_{Nr} \cdot u_r, \tag{2}$$

with f_r as the rule's (crisp) consequence value. The consequence is defined by its parameters $p_{0r}, \ldots, p_{Nr}, p_{jr} \in I\!R$.

We also consider simplified fuzzy models containing only one parameter in the consequences. These models are less general than the models with linear combinations, but they are of interest because it is much easier for humans to interpret them. The initial fuzzy rule R_1 of the simplified fuzzy model has for $I_1 = \emptyset$ the form

$$\texttt{if TRUE then } f_1 = p_0,$$

and has for $I_r \neq \emptyset$ the form

$$\texttt{if } u_{i_{1r}} \texttt{ is } F_{1r} \texttt{ and} \ldots \texttt{and } u_{i_{n_r r}} \texttt{ is } F_{n_r r} \texttt{ then } f_r = p_r, \tag{3}$$

with f_r as the rule's (crisp) consequence value.

In our approach we choose sigmoidal functions as shapes or membership functions for the premises' fuzzy sets F_{jr}. They are defined for $r = 1, \ldots, R$ and $m = 1, \ldots, M$ and $\forall j \in I_r; u_{jm} \in I\!R$ as

$$F_{jr}(u_{jm}) \quad := \quad \frac{1}{1 + e^{\sigma_{jr} \cdot (u_{jm} - \mu_{jr})}}. \tag{4}$$

The parameter μ_{jr} describes the location, parameter σ_{jr} the steepness of the jth membership function in rule r. Figure 2 (right) shows an example of two sigmoidal membership functions with $\mu = 0$ and $\sigma = 5$ (solid line) or $\sigma = -5$ (dashed line).

The rth rule's *strength* w_r for the input vector $\vec{u}_m$ with $m = 1, \ldots, M$; $\vec{u}_m \in \mathcal{U}_1 \times \cdots \times \mathcal{U}_N$; $\mathcal{U}_l \subset I\!R$ is given by

$$w_r(\vec{u}_m) := \bigwedge_{j \in I_r} F_{jr}(u_{jm}) \qquad \text{and by choosing the product as t-norm we get} \qquad w_r(\vec{u}_m) = \prod_{j \in I_r} F_{jr}(u_{jm}). \tag{5}$$

For $r = 1, \ldots, R; \forall \vec{u} \in \mathcal{U}_1 \times \cdots \times \mathcal{U}_N$ we also define the *normalized strength* $v_r(\vec{u})$ as

$$v_r(\vec{u}) := \frac{w_r(\vec{u})}{\sum_{i=1}^{R} w_i(\vec{u})} \qquad \text{where we have} \qquad \sum_{k=1}^{r} v_k(\vec{u}) = 1. \tag{6}$$

So, the normalized strength can be regarded as a possibility distribution.

The fuzzy model performs a mapping $\hat{y} : \mathcal{U}_1 \times \cdots \times \mathcal{U}_N \mapsto \mathcal{Y}$ with $\mathcal{U}_j \subset I\!R$ and $\mathcal{Y} \subset I\!R$. By using the *product inference* (Larsen) as fuzzy inference and *weighted average* for defuzzification, the crisp model output can be calculated as

$$\hat{y}(\vec{u}) \quad = \quad \frac{\sum_{r=1}^{R} w_r(\vec{u}) \cdot f_r(\vec{u})}{\sum_{r=1}^{R} w_r(\vec{u})}. \tag{7}$$

Figure 2 (left) shows the two consequences f_1 and f_2 (solid lines) and the model output $\hat{y}$ (dashed line) of the fuzzy model

$$\begin{aligned} &\mathtt{R_1}: \texttt{ if } u_1 \texttt{ is } F_1 \texttt{ then } f_1 = 1 + 0,5 \cdot u_1 \\ &\mathtt{R_2}: \texttt{ if } u_1 \texttt{ is } F_2 \texttt{ then } f_2 = 1 - 1,5 \cdot u_1 \end{aligned}$$

with $\mu_1 = \mu_2 = 0$, $\sigma_1 = 5$ and $\sigma_2 = -5$ for the membership functions F_1 and F_2, which are shown in Figure 2 (right). One can see how the model output is formed by the two linear consequences f_1 and f_2 with a fuzzy transition between $u_1 = -0.7$ and $u_1 = 0.7$.

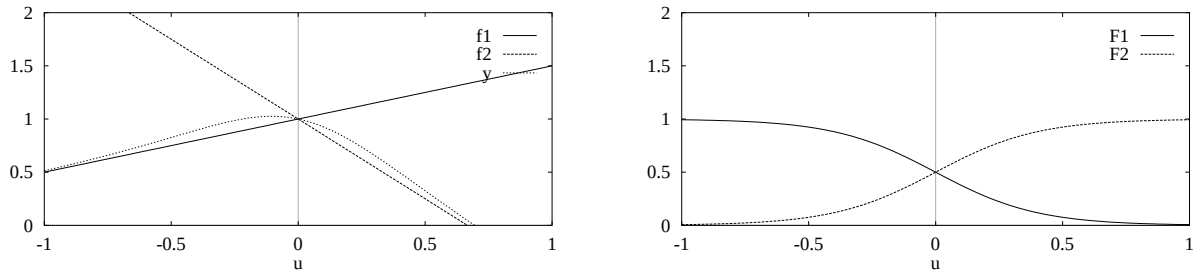

Figure 2: Fuzzy model containing two rules for a one dimensional input.

2.2 Parameter Identification

Adjusting a model's parameters based on a set of training data is also called learning. For the given fuzzy models, we have to optimize all parameters in the rules' premises and consequences. The problem can be divided into two subproblems: optimization of the premise parameters, which is a nonlinear optimization problem, and optimization of the consequence parameters. Optimization of the consequence parameters reduces to a linear optimization problem when fixing the premises' parameters [Sugeno 85, Yen 98].

Parameter optimization is done by minimization of the global modeling error

$$\|\vec{\varepsilon}\|^2 := \|\vec{y} - \vec{\hat{y}}\,\|^2, \tag{8}$$

where $\vec{\hat{y}} := (\hat{y}_1, \ldots, \hat{y}_M)^T$ with $\hat{y}_m := \hat{y}(\vec{u}_m)$ for $m = 1, \ldots, M$.

For the premise parameter optimization one could choose any nonlinear optimization technique. We apply RPROP, a powerful nonlinear optimization technique which was originally developed for artificial neural network training [Riedmiller 93, Zell 94]. We choose RPROP because it is much faster than simple gradient descent methods and it is easier to apply than other sophisticated gradient descent methods as for example Levenberg-Marquardt algorithms [Hagan 94]. In this paper we do not explain in detail how to apply RPROP to the given fuzzy model, since this is already done in [Männle 95, Männle 96].

For optimization of the consequence parameters we first apply singular value decomposition (SVD) which yields a numerically stable solution of a linear least squares optimization problem [Klema 80, Chen 94].

Let the $M \times (N+1) \cdot R$ matrix A be defined as

$$A := \begin{pmatrix} v_1(\vec{u}_1) & v_1(\vec{u}_1) \cdot u_{11} & \ldots & v_1(\vec{u}_1) \cdot u_{N1} & \ldots & v_R(\vec{u}_1) & \ldots & v_R(\vec{u}_1) \cdot u_{N1} \\ \vdots & & & & & & & \\ v_1(\vec{u}_m) & v_1(\vec{u}_m) \cdot u_{1m} & \ldots & v_1(\vec{u}_m) \cdot u_{Nm} & \ldots & v_R(\vec{u}_m) & \ldots & v_R(\vec{u}_m) \cdot u_{Nm} \\ \vdots & & & & & & & \\ v_1(\vec{u}_M) & v_1(\vec{u}_M) \cdot u_{1m} & \ldots & v_1(\vec{u}_M) \cdot u_{NM} & \ldots & v_R(\vec{u}_M) & \ldots & v_R(\vec{u}_M) \cdot u_{NM} \end{pmatrix} \tag{9}$$

with v_r as in (6) and the consequence parameter vector $\vec{p} := (p_{01}\, p_{11} \ldots p_{N1} \ldots p_{0R}\, p_{1R} \ldots p_{NR})^T$ we can write the model output $\vec{\hat{y}}$ of all examples simply as $\vec{\hat{y}} = A \cdot \vec{p}$.

The v_k in A depend on the premise parameters μ and σ. These are considered constant during the optimization of $\vec{p}$. So, the error $\|\vec{\varepsilon}\|^2$ only depends on the parameters p_k and we have for the minimal error:

$$\frac{\partial \|\vec{\varepsilon}\|^2}{\partial p_k} = 0. \tag{10}$$

By choosing the Euclidean norm L_2 for $\|\cdot\|$ we get for $k = 1, \ldots, (N+1) \cdot R$

$$\frac{\partial \|\vec{\varepsilon}\|^2}{\partial p_k} \;=\; \sum_{m=1}^{M} 2 \left(y_q - \sum_{j=1}^{(N+1)\cdot r} a_{mj} p_j \right) (-a_{mk}) \;=\; 0. \tag{11}$$

With (9) we can also write

$$\begin{aligned} 2\left(A\vec{p}-\vec{y}\right)^T A &= 0 \\ A^T A\vec{p} &= A^T\vec{y} \\ \vec{p} &= \left(\left(A^T A\right)^{-1} A^T\right)\vec{y}. \end{aligned} \tag{12}$$

For these linear regression equations, SVD computes U, D, and V so that $A = UDV^T$, where $U^TU = E$, D is diagonal, and V is orthogonal. Using this we get

$$\begin{aligned} \vec{p} &= \left(VD^TU^TUDV^T\right)^{-1} VD^TU^T\vec{y} \qquad (A = UDV^T) \\ &= \left(VD^TDV^T\right)^{-1} VD^TU^T\vec{y} \qquad (U^TU = E), (D^TD \text{ diagonal}) \\ &= \left(D^TD\right)^{-1} VD^TU^T\vec{y} \qquad (V \text{ orthogonal} \Rightarrow VV^T = E) \\ &= VD^{-1}U^T\vec{y}. \end{aligned} \tag{13}$$

Using the above approach for consequence parameter optimization and RPROP for premise parameter optimization at *each* step we did not get good results. The reason is that the linear optimization calculates the best solution for the consequence parameters in one step. After that, the iteratively adapting RPROP did not change the premise parameters any more. Apparently, by applying SVD the parameter optimization of the whole fuzzy model kept sticking in a local minimum. We therefore apply SVD only for the initial linear fuzzy model. For all other models we *use RPROP to iteratively adapt all parameters at the same time*.

2.3 Heuristic Search of Model Structure

The determination of the *optimal* model structure is a combinatorial problem and an efficient algorithm to solve such a problem is not (yet) available. Therefore, a heuristic search algorithm is used to find a *good* but not necessarily optimal structure within a reasonable calculation time.

In this work we use a heuristic search very similar to that described in [Takagi 85, Sugeno 88]. In Figure 3 the heuristic search algorithm is given in pseudo code. It alternately determines a new model structure M_{cand}, then optimizes this structure and finally calculates its quality q_{cand}. In between each epoch R, the best structure of all investigated structures is saved in M_R and becomes the starting point for the next epoch. In every epoch each rule is refined in each input variable, yielding $R \cdot N$ possibilities to examine. Thus, at every epoch the input space is once more divided by adding a new rule.

Figure 4 shows the possibilities of model refinements to examine. The algorithm starts with one rule containing a linear model. In the first epoch, assuming the refinement in the ith variable was the best, we have a model with two rules with premises containing u_i. Then, the model is further refined by examinating all $2 \cdot N$ possibilities yielding to a three-rule model, say $u_i - u_j$, where the refinement in u_j in the second rule was best, and finally the four-rule model $u_i - u_j - u_k$ in the third epoch. As an advantage, the modeling need not start at the one-rule model. One can formulate à priori knowledge by a set of rules and then start the algorithm improving this initial model.

When refining a model, the subspace covered by one rule is devided to be covered by two rules afterwards. The inital partition (given by the fuzzy set's μ) is performed on the middle of the subspace.

The choice of the stopping criterion for the optimization process is crucial for the quality of the model. We use the sum squared error $\|\vec{\varepsilon}\|^2$ (8) to calculate the model quality. In order to avoid "over-learning", i.e., the identification of noise in the training data, we divide the training set into two sets A and B. Then, the RPROP algorithm is performed on A whereas the quality calculation is based on B (cross validation). Thus, identification of noise in A usually yields a decreasing quality since the noise in A and B can be considered different and the training procedure is terminated.

```
R := 1                          /* initial linear model (see sect. 2.1) */
optimize M_1                    /* e.g., through SVD (see sect. 2.2) */
calculate q_1                   /* model quality */
M_opt := M_1
q_opt := q_1
while R < R_max and (not model good enough) do
    R := R + 1
    q_r := ∞
    for r := 1 to R − 1 do
        for j := 1 to N do
            M_cand := M_{R−1}
            M_cand := M_cand + rth rule divided in the jth variable
            optimize all parameters (e.g., using RPROP)
            calculate q_cand
            if q_cand < q_R     /* best model in epoch R so far */
                M_R := M_cand
                q_R := q_cand
            end if
        end for
    end for
    if q_R < q_opt     /* overall best model so far */
        M_opt := M_R
        q_opt := q_R
    end if
end while
```

Figure 3: Scheme of the heuristic structure search.

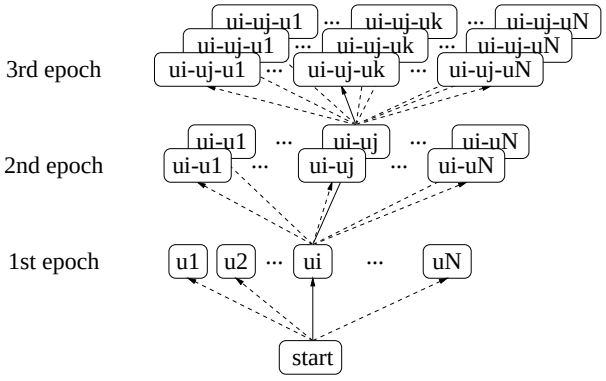

Figure 4: Search tree for input space partitioning.

3 EXAMPLES

In this paper we focus on the heuristic search algorithm and the iterative way the models are more and more refined at each epoch. The examples shown here have two aims:

As a first example we chose a step function in order to show the performance of the RPROP parameter optimization. This type of function is relatively hard to approximate for a model having smooth sigmoidal membership functions, since the optimization technique has to find a very high steepness of the membership functions.

The second example demonstrates how the structuring algorithm and especially its input space partitioning work. For this purpose we chose Gaussian bell functions as a more complex example of a two dimensional mapping.

The evaluation of the algorithm's performance is described briefly, since you can find two examples in [Männle 95, Männle 96].

3.1 Step Function

Step functions are quite hard to approximate by models having smooth sigmoidal membership functions. Crisp functions are usually hard to approximate by fuzzy models. This is because the optimization algorithm has to adjust the membership function to be very steep, i.e., has to find very high values for σ.

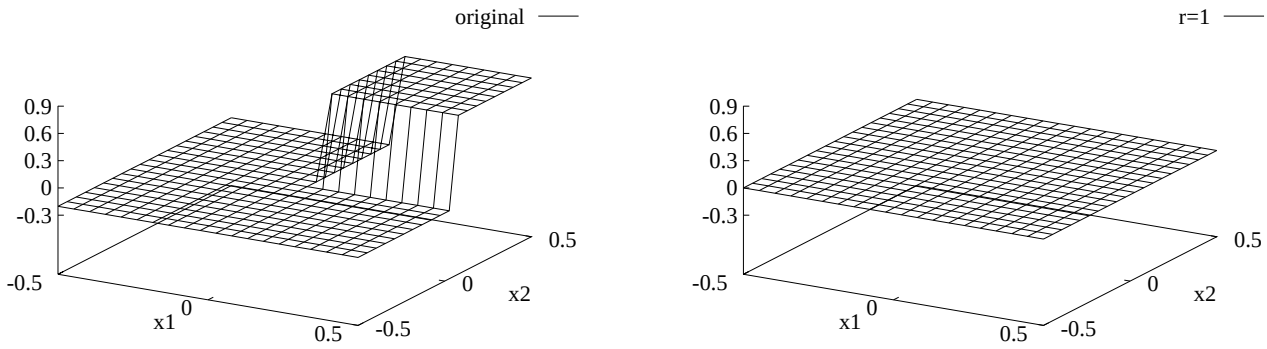

Figure 5: Original step function (left) and model containing one rule (right).

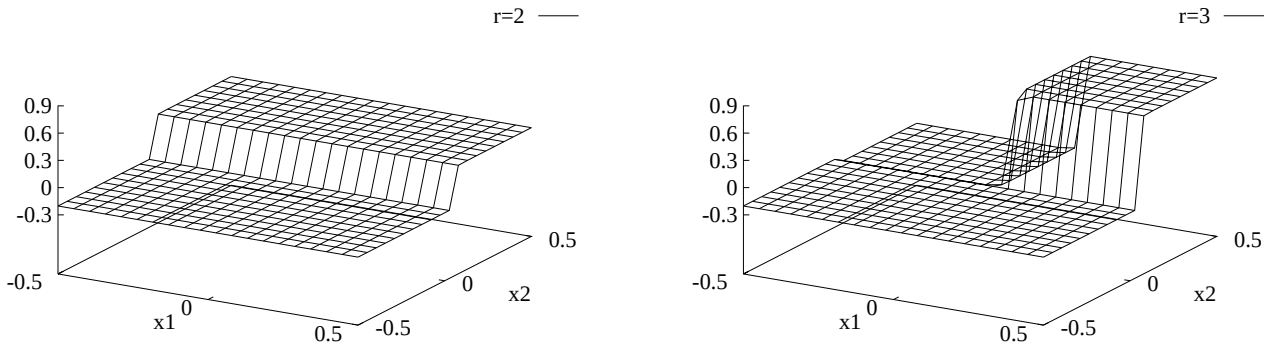

Figure 6: Models for step function containing two (left) and three rules (right).

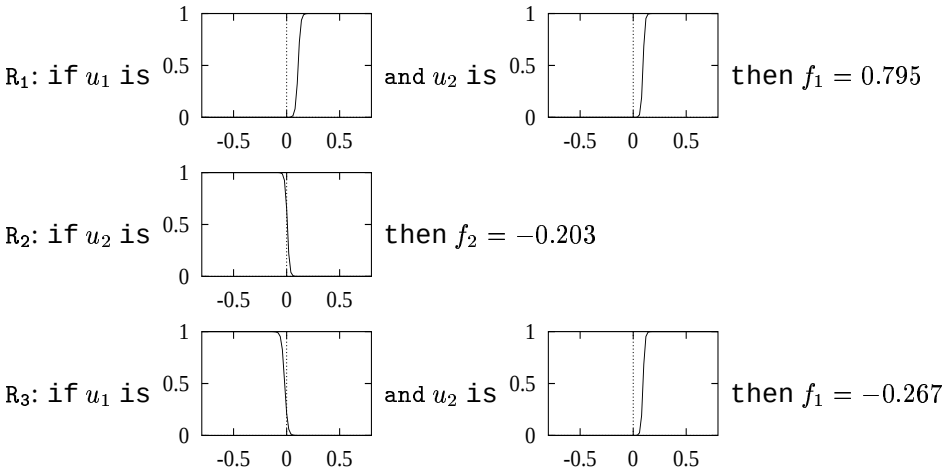

Figure 7: Model for step function, three rules.

Figure 5 (left) shows the original step function. From figure 5 (right) to figure 6 you can see the development of the fuzzy

model. As expected, a model containing three rules completely describes the original function

In figure 7 you find the resulting fuzzy model. All μ are adjusted to the step which lies between 0 and 0.05. All σ are adjusted to values about 100 yielding the very steep membership functions which can be seen in figure 7.

To find the resulting model, six models had to be examined. In the first epoch two possible refinements ($N = 2$) of the initial rule, in the second epoch $N \cdot R = 2 \cdot 2 = 4$ possibilities. To optimize the models' parameters RPROP needed on the average only about 16 iterations for each model, so that computation time of our C++ implementation on a Sun UltraSparc1 was only a few seconds.

3.2 Two Gaussian Bell Functions

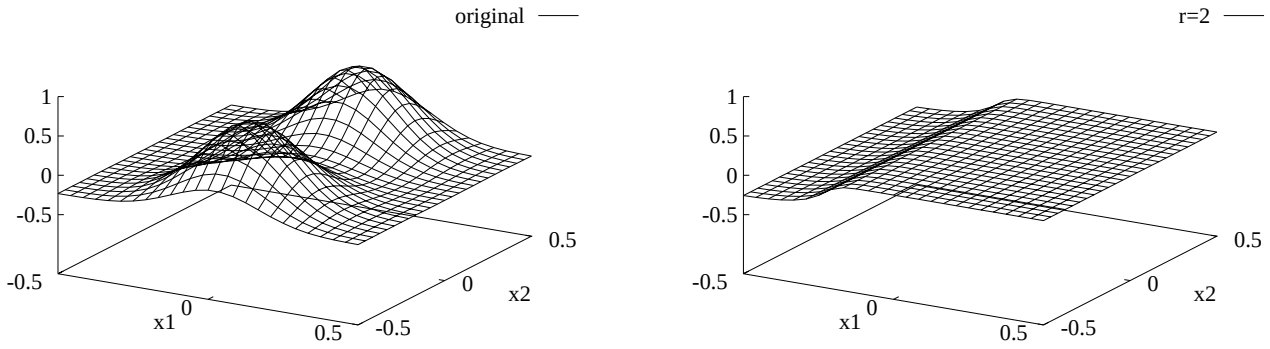

Figure 8: Original Gaussian bell functions (left) and model containing two rules (right).

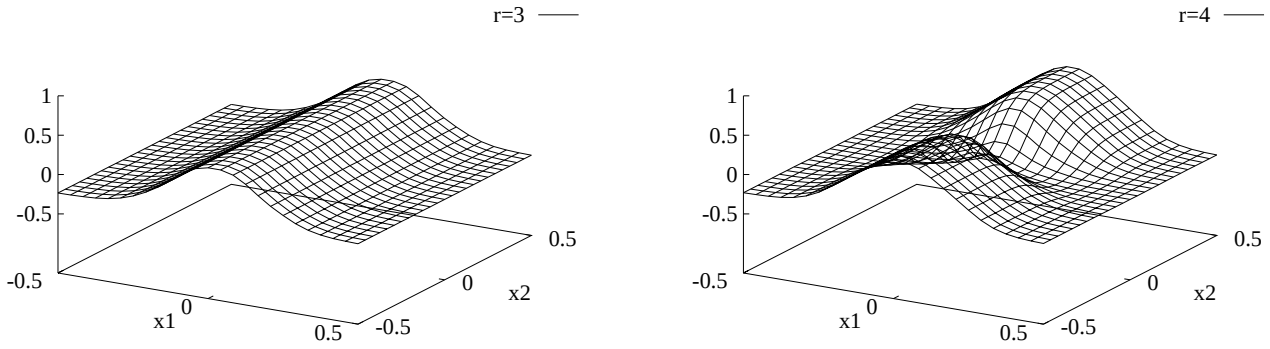

Figure 9: Models for Gaussian bell functions containing three (left) and four rules (right).

To visualize the way the modeling algorithm works, we chose a two dimensional function as an example. The function to be approximated consists of two Gaussian bell functions in the two dimensional input space. From this function, 25x25 equidistant and normalized base points were generated. (see figure 8 (left)). For better comprehension of the resulting models we chose the simplified fuzzy models containing rules with constant consequences.

The figures 8 (right), 9, and 10 show the development of the fuzzy model. The figures 11 to 13 show the found input space partitioning (grey bars) and the investigated refinements (dashed lines) for the first six epochs. The dashed lines also indicate the initial location of the new fuzzy sets before optimization. The fuzzy models of the first six epochs are given in the figures 14 to 18 in the appendix.

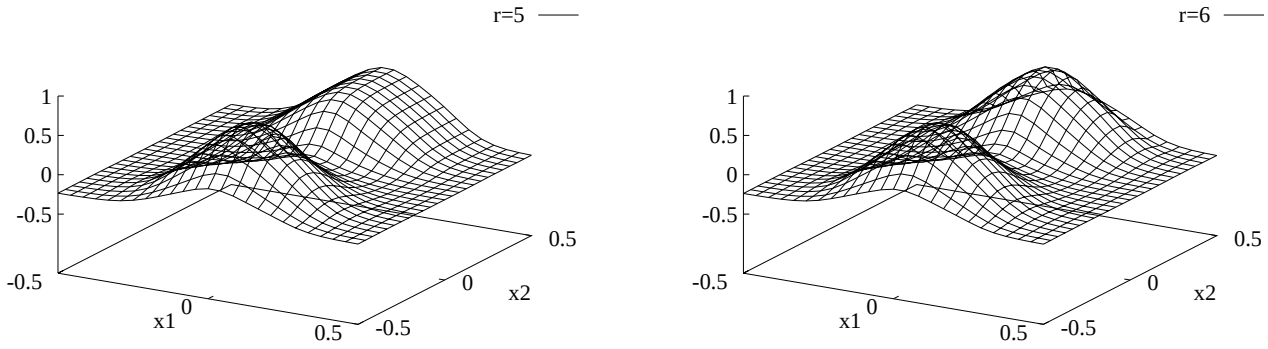

Figure 10: Models for Gaussian bell functions containing five (left) and six rules (right).

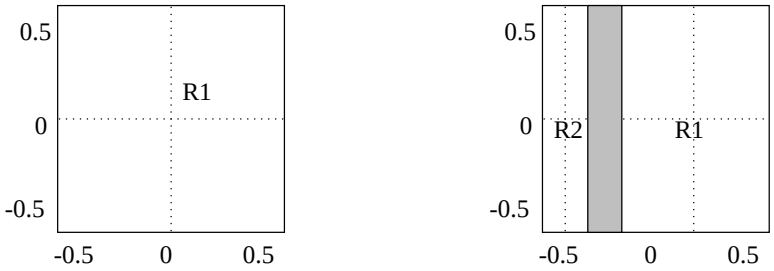

Figure 11: Input space partition, one and two rules.

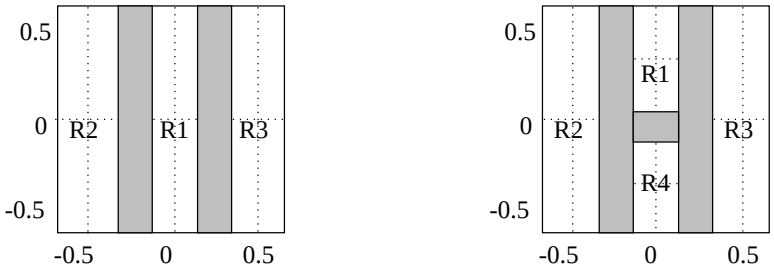

Figure 12: Input space partition, three and four rules.

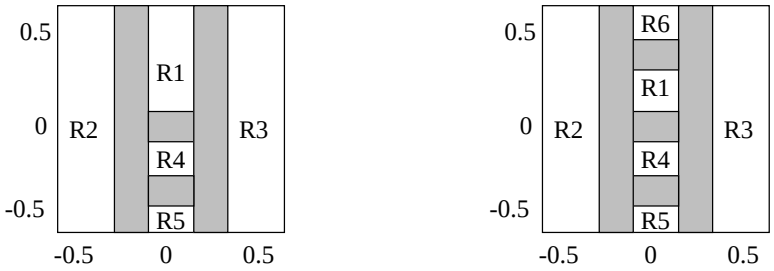

Figure 13: Input space partition, five and six rules.

In figure 8 (right) we can see the two partial models: one rule with $\hat{y} \approx -0.26$ for "u_1 small" ($\mathtt{R}_2$ from figure 14) and the other with $\hat{y} \approx 0.07$ for "u_1 big" ($\mathtt{R}_1$ from figure 14) and a fuzzy transition between them. Rule $\mathtt{R}_1$ is being refined into two new rules. In figure 12 (left) you can see the resulting input space partitioning.

The model of the estimation in figure 9 (right) is listed in figure 16. This model can be interpreted easily. For u_1 small ($\mathtt{R}_2$) and u_1 big ($\mathtt{R}_3$) we have $f_2 \approx f_3 \approx -0.23$. The two "hills" in the middle of the u_1-range are formed by the rules $\mathtt{R}_1$ and $\mathtt{R}_4$ by their positive consequence values.

Figure 10 shows the last refinements rounding off the bell functions. The final input space partition is shown in figure 13 (right), the complete fuzzy model is given in figure 18. Thus, in this example, the original function is modeled with only six fuzzy rules.

3.3 Performance Evaluation

A comparison with four other nonlinear modeling algorithms taken from [Frank 95] is given in [Männle 96]. In some cases of the 40 models to identify the presented approach was outperformed by the best of the other algorithms, but it shows itself as the overall best technique. Especially when approximating the real data it outperformed all other regarded techniques. In [Männle 96] we also examined Box/Jenkins' gas furnace [Box 76] as a well known example for a dynamical system. For this data, a simple model containing only three rules already yields a satisfying approximation.

4 CONCLUSIONS

In this paper we focused on the structure modeling and input partitioning part of our approach to build rule-based fuzzy models. The initial one-rule model is iteratively refined by adding rules, i.e., further partitioning the input space. Every rule can be considered as a local model and the imprecision of the fuzzy sets results in smooth transitions between these local models. We illustrated this by means of a two dimensional example in section 3.2.

We applied RPROP as a powerful optimization technique to the parameter optimization problem. We found it to be robust, i.e., working fine with its standard parameters, and converging very fast to a good minimum (see section 3.1). This makes it possible to built also models for high dimensional training sets (e.g., $N = 10$) that comprise many training examples.

The solution for finding a good model structure must be considered provisionary and it seems worth spending additional effort in developing a better method.

Computation time grows quadratically when rising N linearly. This is the result of the heuristic search algorithm. A stronger heuristic can further reduce this factor. For example by excluding input variables which are not involved in premises during several epochs (potentially indicating that this variable does not contribute to the model and is unimportant), some branches of the search tree (figure 4) are cut. This can considerably reduce the search space. However, practical investigations showed that variables, although seeming unimportant at the beginning, are sometimes needed late in the modeling procedure to get a good model.

Another promising approach is to iteratively refine the model in those input space regions where the highest approximation error ε^2 is made. Note, that one must regard *regions* since regarding *single* input-output pairs can cause a too big influence of possible outliers.

The problem of structure modeling as it appears in this work is well suited for meta algorithms like for example genetic algorithms. Tanaka et al. [Tanaka 94] investigated the use of a genetic algorithm for fuzzy models with trapezoidal membership functions and a restricted model structure. When trying this, research effort should be spent to the coding scheme, since the application of an appropriate coding scheme largely determines the success of such a genetic algorithm.

Finally, one could think of a "top down" approach instead of a "bottom up" method. For a top down approach one chooses a fine partition containing many rules at the beginning and then deletes the rules found not to be necessary. This is done for example in [Jang 93] and [Yen 98]. We did not investigate this approach since it has a crucial drawback when applying it on some of our data sets: We found training data sets of real technical processes to often have an input dimension of 10 or even higher. Even when only choosing a coarse initial partition and dividing each input dimension by only four fuzzy sets, the resulting models become infeasible, since you have to deal with $4^{10} \approx 1$ Mio initial fuzzy rules. Even if you can build a model and reduce the number of fuzzy rules there are usually so many rules left that it is very hard to interpret the model. We found that if you want to *build models automatically* (including parameter optimization *and* structure modeling) for a *high dimensional training set* you usually have no choice but a *bottom up approach*.

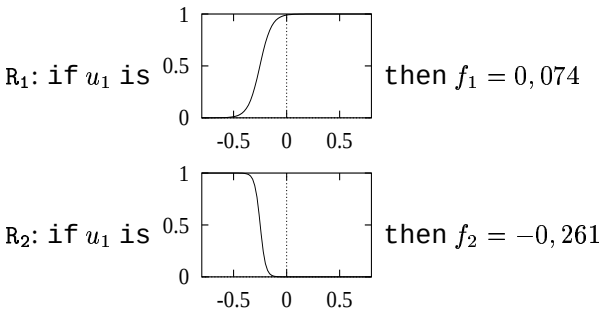

Figure 14: Model for two Gaussian bell functions, two rules.

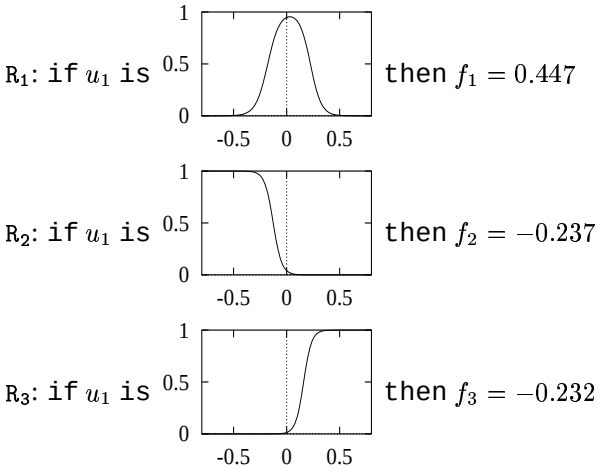

Figure 15: Model for two Gaussian bell functions, three rules.

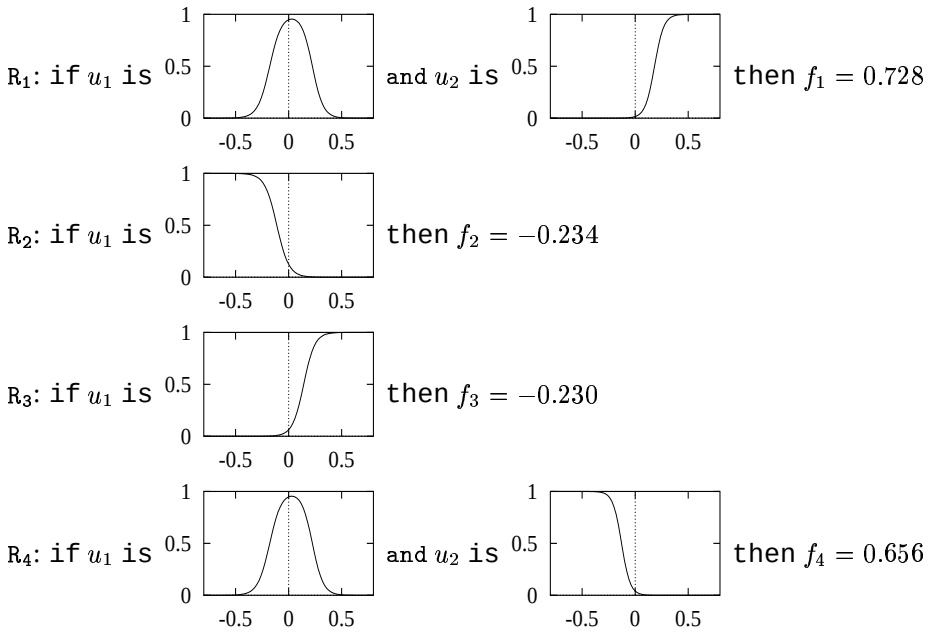

Figure 16: Model for two Gaussian bell functions, four rules.

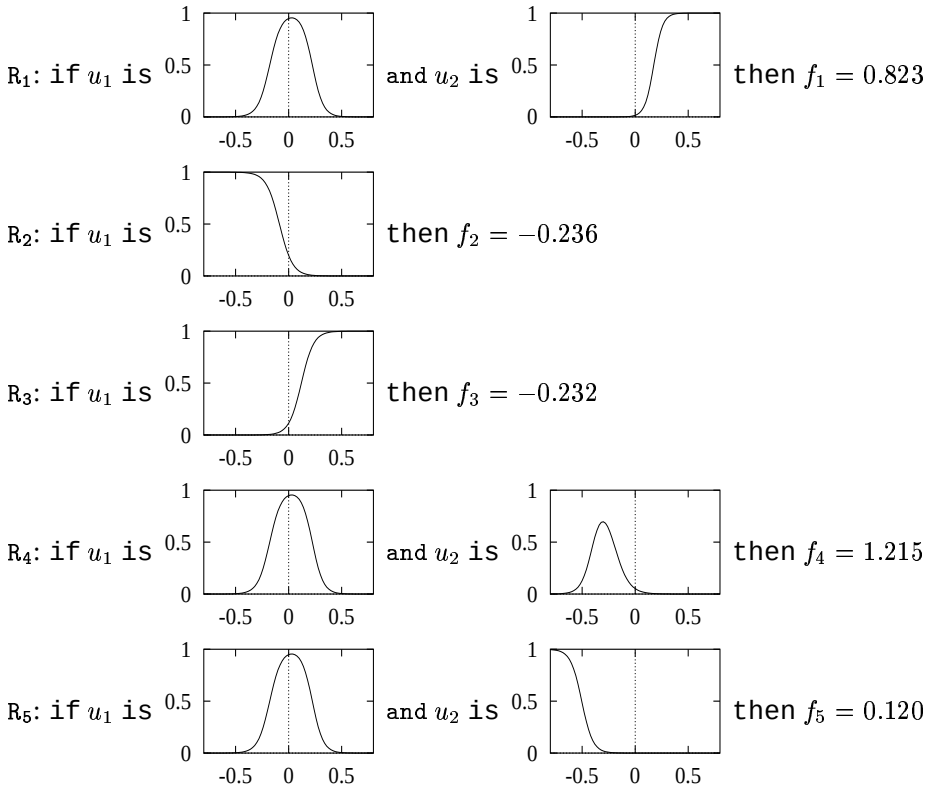

Figure 17: Model for two Gaussian bell functions, five rules.

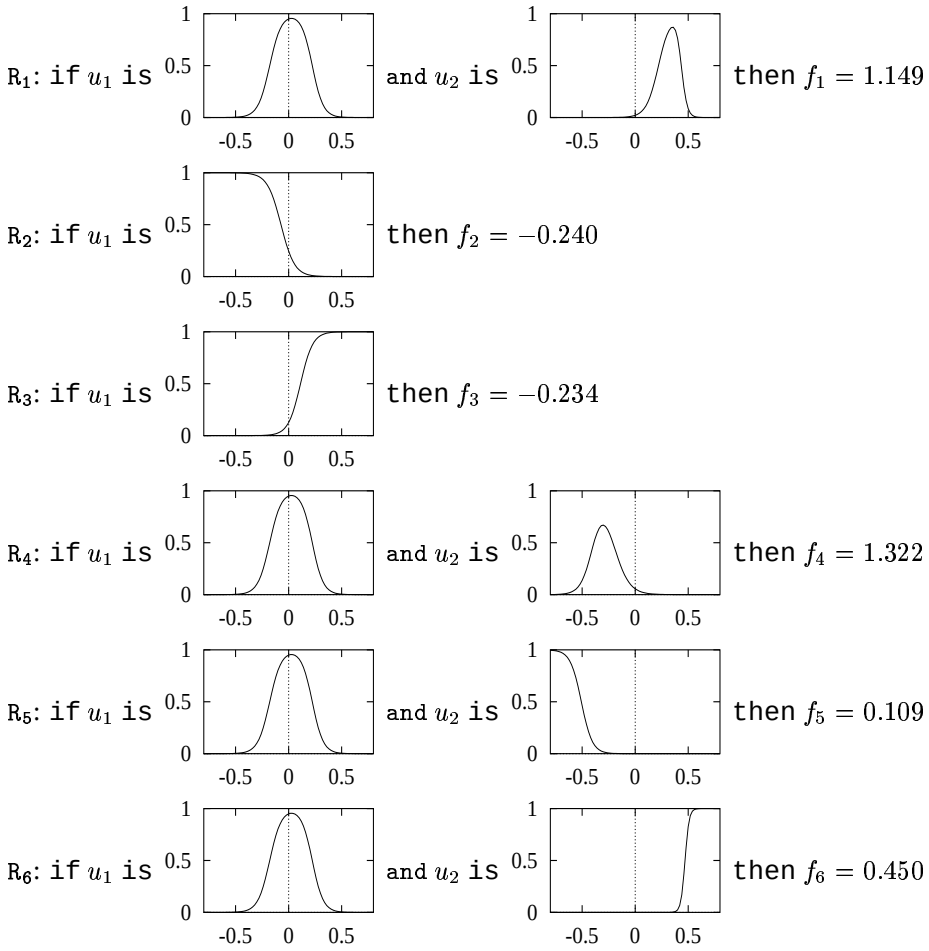

Figure 18: Model for two Gaussian bell functions, six rules.

REFERENCES

Böhler, A. 1994. Lösung von Fuzzy-Relationen-Gleichungen. Diplomarbeit, French-German Institute for Automation and Robotics (IAR), Universität Karlsruhe, Germany.

Box, G.; Jenkins, G. 1976. Time Series Analysis, Forecasting and Control. Holden-Day, 6th edition.

Chen, J.; Lu, J.; Chen, L. 1994. An on-line identification algorithm for fuzzy systems. Fuzzy Sets and Systems, Vol. 64, No. 1, pp. 63–72.

Frank, I. 1995. Modern Nonlinear Regression Methods. Chemometrics and Intellig. Laboratory Syst., Vol. 27, pp. 1–19.

Hagan, M.; Menhaj, M. 1994. Training Feedforward Networks with the Marquardt Algorithm. IEEE Transactions on Neural Networks, Vol. 5, pp. 989–993.

Jang, J.-S. R. 1993. ANFIS: Adaptive-Network-Based Fuzzy Inference System. IEEE Transactions on System, Man, and Cybernetics, Vol. 23, No. 3, pp. 665–685.

Klema, V.; Laub, A. 1980. The singular value decomposition: Its computation and some applications. IEEE Transactions on Automatic Control, Vol. 25, No. 2, pp. 164-176.

Männle, M. 1995. Modellierung Nichtlinearer Systeme mit Unscharfen Regeln. Diplomarbeit, French-German Institute for Automation and Robotics (IAR), Universität Karlsruhe, Germany.

Männle, M.; Richard, A.; Dörsam T. 1996. Identification of Rule-Based Fuzzy Models Using the RPROP Optimization Technique. In Proceedings of Fourth European Congress on Intelligent Techniques and Soft Computing (EUFIT'96), pp. 587–591.

Nakamori, Y.; Ryoke, M. 1994. Identification of fuzzy prediction models through hyperellipsoidal clustering. IEEE Transactions on Systems, Man, and Cybernetics, Vol. 24, No. 8, pp. 1153–1173.

Pedrycz, W. 1991. Processing in relational structures: Fuzzy relational equations. Fuzzy Sets and Systems, Vol. 40, pp. 77-106.

Postlethwaite, B. 1991. Empirical comparison of methods of fuzzy relational identification. IEEE Proceedings-D, Vol. 139, No. 3, pp. 199–206.

Riedmiller, M.; Braun, H. 1993. A Direct Adaptive Method for Faster Backpropagation: The RPROP Algorithm. In Proceedings of the IEEE Int. Conf. on Neural Networks (ICNN), pp. 586–591.

Sugeno, M.; Kang, G. 1988. Structure Identification of Fuzzy Model. Fuzzy Sets and Systems, Vol. 26, pp. 15–33.

Sugeno, M.; Yasukawa, T. 1993. A fuzzy-logic-based approach to qualitative modeling. IEEE Transactions on Fuzzy Systems, Vol. 1, No. 1, pp. 7–31.

Takagi, T.; Sugeno, M. 1985. Fuzzy Identification of Systems and its Application to Modeling and Control. IEEE Transactions on Systems, Man, and Cybernetics, Vol. 15, pp. 116–132.

Tanaka, M.; Ye, J.; Tanino, T. 1994. Identification of Nonlinear Systems Using Fuzzy Logic and Genetic Algorithms. SYSID, Vol. 1, pp. 301–306.

Vrba, J. 1992. Peak-pattern concept and and max-min inverse problem in fuzzy control modeling. Fuzzy Sets and Systems, Vol. 47, No. 1, pp. 1–11.

Yager, R.; Filev, D. 1993. Unified structure and parameter identification of fuzzy models. IEEE Transactions on Systems, Man, and Cybernetics, Vol. 23, No. 4, pp. 1198–1205.

Yen, J; Wang, L. 1998. Improving the Interpretability of TSK Fuzzy Models by Combining Global Learning and Local Learning. IEEE Transactions on Fuzzy Systems, Vol. 6, No. 4, pp. 530–537.

Zadeh, L. 1965. Fuzzy Sets. Information and Control, Vol. 8, pp. 338–353.

Zadeh, L. 1973. The Birth and Evolution of Fuzzy Logic. International Journal on General Systems, Vol. 17, pp. 95–105.

Zadeh, L. 1994. The Role of Fuzzy Logic in Modeling, Identification and Control. Modeling, Identification, and Control, Vol. 15, pp. 191–203.

Zell, A.; Mache, N.; Sommer, T. et al 1994. Stuttgart Neural Network Simulator, Users Manual, Version 3.2. University of Stuttgart, Germany.