A Rule-Based Fuzzy Model for Nonlinear System Identification

Manfred Männle, Alain Richard, Thomas Dörsam

Abstract— **This article discusses a rule-based fuzzy model for the identification of nonlinear MISO (multiple input, single output) systems. The discussed method of fuzzy modeling consists of two parts: structure modeling, i.e. determing the number of rules and input variables involved respectively, and parameter optimization, i.e. optimizing the location and form of the curves which describe the fuzzy sets. For structure modeling we use a modified TSK-model. The TSK-model was first proposed by Takagi, Sugeno, and Kang in [1], [2]. For parameter optimization we propose the use of RPROP, a powerful optimization technique originally designed for Neural Network training (see also [3], [4]). We applied RPROP to the modified version of the TSK-model, implemented the algorithm, and tested its performance [5], [6]. In this article we focus on the structure modeling part and show by an example how this structuring algorithm performs an input space partitioning.**

Keywords— **Rule-based fuzzy model, nonlinear system identification, input space partitioning, parameter optimization, RPROP.**

I. Introduction

Fuzzy theory finds its application to deal with problems in which there is imprecision caused by the absence of sharp criteria [7]. This is for example in human language. Linguistic terms can be defined by fuzzy sets and we can formulize fuzzy if-then rules [8]. 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 take another 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 [9].

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 want here is an interpretable model, i.e. a qualitative description of the underlying process. For this, fuzzy modeling is 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 [10], [11], [12]. Some comparisons can for example be found in [13], [14].

Another approach is to divide the input-output space into *clusters*. Fuzzy rules are generated by projection of these cluster 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 [15], Nakamori and Ryoke use fuzzy c-varieties [16].

In this work we have to deal with another problem. We want to identify complex systems with a *a large number of input variables*. Therefore, methods yielding a polynomial or even exponential number of rules (depending from the number of input variables) are not applicable. That is why we take a "bottom up" rule-based approach: the

Manfred Männle was with the French-German Institute for Automation and Robotics (IAR), Universität Karlsruhe, Germany. He is now with the Institut für Rechnerentwurf und Fehlertoleranz, Universität Karlsruhe, maennle@ira.uka.de.

Alain Richard is with the Centre de Recherche en Automatique de Nancy (CRAN), CNRS URA 821, Université Henri Poincaré, Nancy 1, France, arichard@cran.u-nancy.fr.

Thomas Dörsam is with the Institut für Prozeßrechentechnik und Robotik, Universität Karlsruhe, doersam@ira.uka.de

This work was supported by the IAR Karlsruhe/Nancy and by a European ERASMUS grant.

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 [9] which was first proposed by Takagi, Sugeno, and Kang in [1], [2]. Other derivations from TSK can be found for example in [17], [18].

In section 2 we describe the modified TSK-model we use for fuzzy modeling. We also explain the algorithm's search for the model structure. For a description of RPROP, the parameter optimization, and the necessary calculations see [5], [6]. In chapter 3 we demonstrate the way the algorithm works and performs an input space partitioning. We show this at an example of two-dimensional Gaussian bell functions. Chapter 4 closes this article by some conclusions and perspectives.

II. Fuzzy identification

In this paper we perform a fuzzy model identification by approximating m measured or simulated input-output data pairs $(\vec{u}_1, y_1), \ldots, (\vec{u}_m, y_m)$ with $\vec{u}_i \in I\!R^N$, $y_i \in I\!R$ (MISO system with N input variables). For $i = 1, \ldots, m$, the regarded fuzzy model performs a mapping

$$\hat{y}_i = \hat{f}(\vec{u}_i) \overset{!}{\approx} y_i. \tag{1}$$

A. The fuzzy model

We use RPROP for the optimization of the membership functions' parameters. RPROP belongs to the family of gradient decent algorithms and we therefore need derivable membership functions. We choose the following sigmoid membership functions since they come near to the triangular membership functions of the original TSK-model and their derivations can be easily calculated. A triangular like $\bigwedge$ is imitated by the superposition of an ascending and a descending sigmoid. Regarding this, we define the fuzzy sets as

$$F_{jk}(u_{ji}) \;:=\; \frac{1}{1 + e^{\sigma_{jk} \cdot (u_{ji} - \mu_{jk})}}, \tag{2}$$

with the parameters μ and σ to be optimized.

The index set $I_k = \{i_{1k}, \ldots, i_{n_k k}\}$ with indices $i_{jk} \in \{1, \ldots, N\}$ is used to specify the variables involved in the kth rule's premise. This is necessary, since not all input variables need to appear in each premise.

As usual for TSK-like models, the rules' consequences are weighted sums of the input variables $p_{0k} + p_{1k} \cdot u_1 + \ldots + p_{Nk} \cdot u_N$, defining an arbitrary hyperplane in the input-output space. We name this form "full" consequences. We also consider a simpler class of consequences, the "constant" consequence p_{0k}, consisting only of one input variable independent parameter. This yields a hyperplane which is orthogonal to the output dimension (y-axis). Therefore, the form of the used fuzzy rules is

$$\mathtt{R_k}\text{:} \begin{cases} \texttt{if } u_{i_{1k}} \texttt{ is } F_{1k} \texttt{ and } \ldots \texttt{ and } u_{i_{n_k k}} \texttt{ is } F_{n_k k} \\ \texttt{then } f_k = p_{0k} \underbrace{+p_{1k} \cdot u_1 + \ldots + p_{Nk} \cdot u_N}_{\text{optional}}. \end{cases} \tag{3}$$

For $I = \emptyset$ we have a special case. We then have a linear model

$$\mathtt{R_k}\text{:} \begin{cases} \texttt{if TRUE} \\ \texttt{then } f_k = p_{0k} \underbrace{+p_{1k} \cdot u_1 + \ldots + p_{Nk} \cdot u_N}_{\text{optional}}. \end{cases} \tag{4}$$

The *fuzzy model* M_R is given by a set of r fuzzy rules, i.e. $M_R = \{R_1, \ldots, R_r, I_1, \ldots, I_r\}$.

The membership w_k of $\vec{u}_q$ to the rule R_k ($k = 1, \ldots, r$) is

$$w_k(\vec{u}_q) \;:=\; \bigwedge_{j \in I_k} F_{jk}(u_{jq}) \tag{5}$$

and by choosing the *product* as t-norm we obtain

$$w_k(\vec{u}_q) \;=\; \prod_{j \in I_k} F_{jk}(u_{jq}). \tag{6}$$

The model output is calculated via *product inference* (Larsen) and *weighted average* by

$$\hat{y}(\vec{u}) \;=\; \frac{\sum_{k=1}^{r} w_k(\vec{u}) \cdot f_k(\vec{u})}{\sum_{k=1}^{r} w_k(\vec{u})}. \tag{7}$$

This fuzzy model is a composition of several linear model parts with fuzzy transitions. As an example, figure 1 (top) shows the output $\hat{y}$ (dotted line) of the fuzzy model given by

$\mathtt{R_1}$: if u_1 is F_1 then $f_1 = 1 + 0.5 \cdot u_1$
$\mathtt{R_2}$: if u_1 is F_2 then $f_2 = 1 - 1.5 \cdot u_1$,

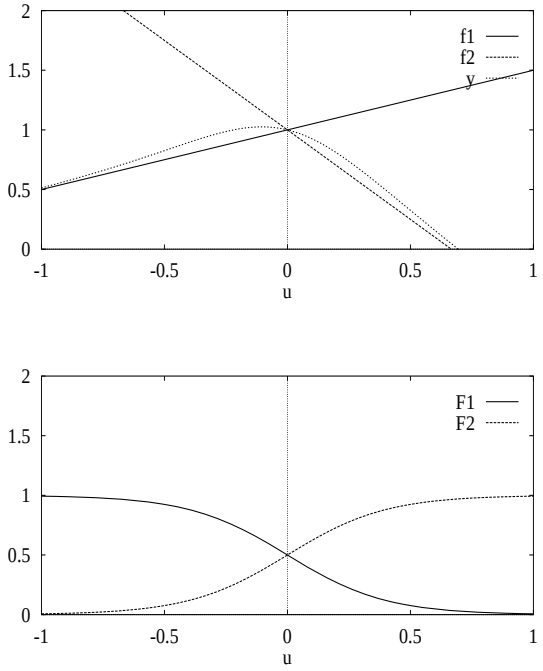

Fig. 1. One-dimensional example of a fuzzy model with two rules.

as the fuzzy transition between the two linear model parts f_1 and f_2. The transition is defined by the fuzzy sets F_1 and F_2 with the parameters $\mu_1 = \mu_2 = 0,\ \sigma_1 = 5,$ and $\sigma_2 = -5$, see figure 1 (bottom).

B. Parameter identification

Parameter identification is performed by minimization of the prediction 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 $\hat{y}_i := \hat{y}(\vec{u}_i)$.
The error depends from the model output, i.e. for a *given* model structure it only depends from the fuzzy rules' parameters. Therefore, using this error function, all parameters can be optimized by using for example gradient decent algorithms.
The nonlinear optimization algorithm RPROP belongs to the family of gradient descent algorithms with variable step. It changes its behavior independently for every parameter to be optimized regarding the local topology of the energy or error function. For a general description see [3], [4]. Further remarks concerning its application to our optimization problem can be found in [5], [6].
The parameter and step updates only depend from the gradient's *sign* and not from the gradient's *magnitude*, which makes the algorithm robust against undesirable magnitude fluctuations. To obtain more information, the sign changes are regarded, which showed itself as a reliable indicator for the error function's local topology.
To apply RPROP to the given optimization problem, at every iteration t the partial derivations of $\|\vec{\varepsilon}^t\|^2$ by p_{jk} (for $j = 0, \ldots, N;\ k = 1, \ldots, r$) and by μ_{jk} and σ_{jk} (for all $j \in I_k,\ k = 1, \ldots, r$) are needed. See [5], [6] for the necessary derivations.

C. Heuristic search of model structure

We use a heuristic search algorithm since the determination of the *optimal* model structure is a combinatorial problem and an efficient algorithm to solve such a problem is not (yet) available. By using a heuristic search we endeavor to find a *good* but not necessarily optimal structure within a reasonable calculation time.

```
r := 1          /* initial linear model (see ch. II-A) */
optimize M_1 /* e.g. singular value decomposition */
calculate c_1  /* model quality */
M_opt := M_1
c_opt := c_1
while r < r_max and (not model good enough) do
  r := r + 1
  c_r := ∞
  for q := 1 to r − 1 do
    for j := 1 to N do
      M_cand := M_{r−1}
      M_cand := M_cand + qth rule divided in
                the jth variable
      optimize all parameters (e.g. with RPROP)
      calculate c_cand
      if c_cand < c_r    /* M_cand better than M_r */
        M_r := M_cand
        c_r := c_cand
      end if
    end for
  end for
  if c_r < c_opt    /* M_r better than M_opt */
    M_opt := M_r
    c_opt := c_r
  end if
end while
```

Fig. 2. Scheme of the heuristic structure search.

In Figure 2 the heuristic search algorithm is given

in a pseudo programming language. It alternately determines a new model structure and then optimizes this structure. In between each epoch, the best structure of all investigated structures is saved and becomes the starting point for the next epoch. In every epoch r, 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.

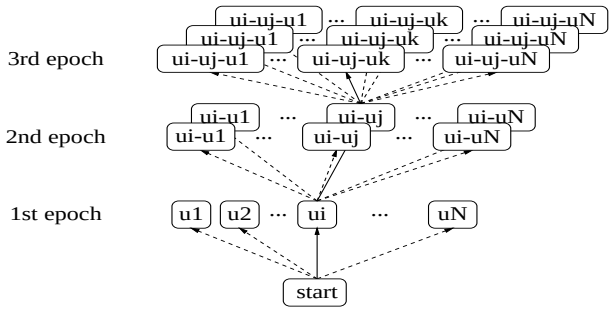

Fig. 3. Search tree for input space partitioning.

Figure 3 shows the possibilities of model refinements to examine. The algorithm starts with one rule containing a linear model. At 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 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 $u_i - u_j - u_k$ at the third epoch. As an advantage, the modeling need not start at the one-rule model. One can formulate a priori knowledge by a set of rules and then start the algorithm improving this initial model.
Starting from scratch, the algorithms needs

$$\begin{aligned} T(r_{max}, N, m, I) &= \mathrm{O}\left(\sum_{r=1}^{r_{max}} r \cdot N \cdot (rNmI)\right) \\ &= \mathrm{O}\left(N^2 mI \cdot \frac{r_{max}(r_{max}+1)(r_{max}+2)}{6}\right) \\ &= \mathrm{O}\left(r_{max}^3 N^2 mI\right) \end{aligned} \tag{9}$$

calculation steps[1] in the worst case. In (9), I is the maximal number of RPROP iterations. In practical applications we found $I = 100$ being sufficient as a maximum border. At a first view, the cubic calculation time increase in r_{max} seems to be critical for the calculation time, but since there are usually not many rules necessary (between five and fifteen), one need not to worry about this. A positive and more important aspect is the only linear growth in the number of training examples m.
We use the R^2 criterion to calculate the model quality c. It is defined by

$$R^2 = 1 - \frac{V(\varepsilon)}{V(y)} = 1 - \frac{\sum_{i=1}^{n}(\varepsilon_i - E(\varepsilon_i))^2}{\sum_{i=1}^{n}(y_i - E(y_i))^2} \tag{10}$$

with the variance $V(\cdot)$ and the expectation value $E(\cdot)$. R^2 is to be maximized. 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 like in [1]. Then, the RPROP algorithm is performed on A while the quality calculation is based on B. Thus, identification of noise in A usually yields a decreasing quality since the noise in A and B can be considered different. As soon as the model quality starts decreasing, the training procedure is interrupted.

III. Examples

The first example demonstrates how the algorithm, especially the input space partitioning, works. For this purpose we chose a simple two-dimensional example. The evaluation of the algorithm's performance is described briefly since it is already described by two examples in [5], [6].

A. Two Gaussian bell functions

To visualize the way the modeling algorithm works, we chose the following two-dimensional 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 4). For better comprehension of the resulting models we chose rules with constant consequences.
The figures 5 to 10 show the development of the fuzzy model. In figure 5 we can see the two partial models: one rule with $\hat{y} \approx -0.23$ for u_1 small and the other with $\hat{y} \approx 0.06$ for u_1 big and a fuzzy transition between these two partial models. In

[1]For the definition and some theorems of the O-calculus see [19] pp. 26–31.

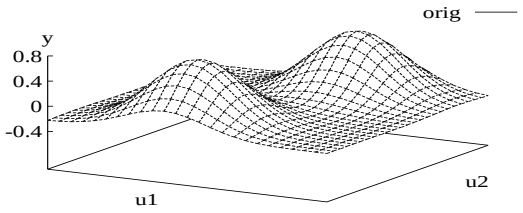

Fig. 4. Two Gaussian bell functions.

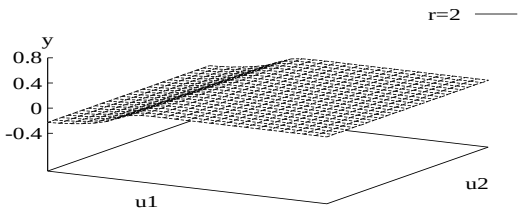

Fig. 5. Estimation, constant consequences, $r = 2$.

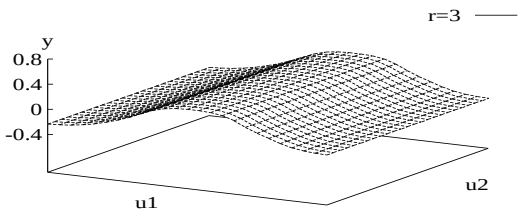

Fig. 6. Estimation, constant consequences, $r = 3$.

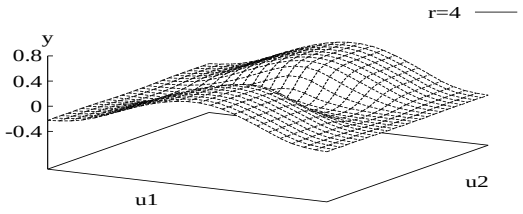

Fig. 7. Estimation, constant consequences, $r = 4$.

figure 6, the rule for u_1 big is refined into two new rules. The models of the estimations in figures 5 to 7 are listed in the figures 11 to 13. For example the model in figure 13 can be interpreted easily: for u_1 small ($\mathtt{R}_2$) and u_1 big ($\mathtt{R}_3$) we have

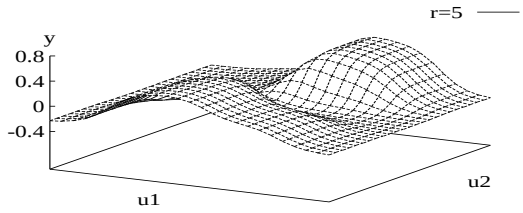

Fig. 8. Estimation, constant consequences, $r = 5$.

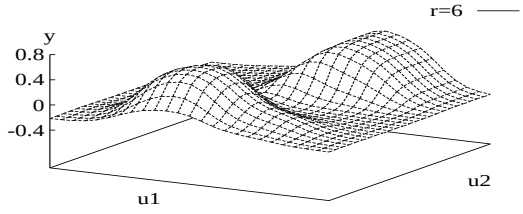

Fig. 9. Estimation, constant consequences, $r = 6$.

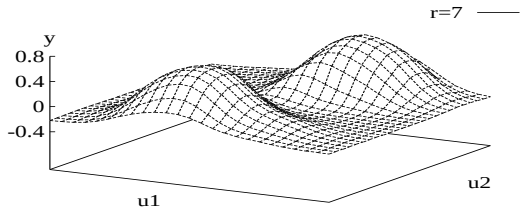

Fig. 10. Estimation, constant consequences, $r = 7$.

$f_2 \approx f_3 \approx -0.2$. 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. The next refinement (figure 8) pays regard to the displacement of the two bell functions. (The two peaks are located at different u_2 *and* u_1.) Finally, the additional rules in figure 9 and figure 10 round off the bell functions. Thus, in this example, the original function is modeled with seven fuzzy rules.

B. Performance evaluation

A comparison with four other nonlinear modeling algorithms is given in [5]. 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. Espe-

cially when approximating the real data it outperformed all other regarded techniques. In [5] we also examined Box/Jenkins' gas furnace [20] as a well known example for a dynamical system. For this data, a simple model containing only three rules already yields a satisfying approximation.

IV. Conclusion

In this paper we describe the structure modeling and input partitioning of our approach of rule-based fuzzy modeling. The initial one-rule model is iteratively refined by adding rules and partitioning the input space. Every rule can be considered as a local model and the rules' fuzziness results in smooth transitions between these local models. We illustrated this by a two-dimensional example.

Whereas we applied a powerful optimization technique to the parameter optimization problem (see [5], [6]), the solution for finding a good model structure must be considered provisionary and it seems worth spending additional effort in developing a better method.

In (9), the quadratic growth of $\mathrm{O}(\cdot)$ in N 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 3) 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.

Finally, 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. [18] investigated the use of a genetic algorithm for fuzzy models with trapezoidal membership functions and a restricted model structure. A similar approach can be taken here. All conditions for the application of a genetic algorithms to our method are satisfied: The model structure can be coded as a string, consequence parameters are adapted independently from input partitioning, and there is an energy or cost function ε^2 which can be calculated rather quickly. 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.

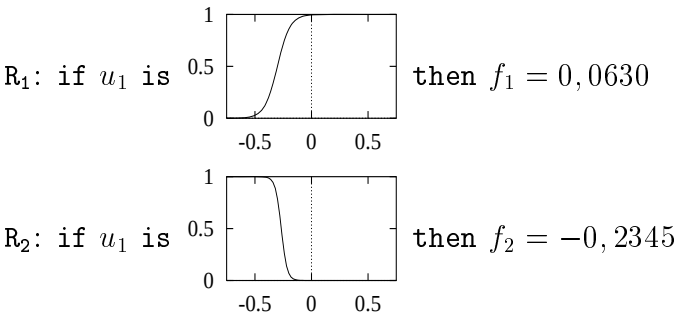

Fig. 11. Model for two Gaussian bell functions, (constant consequences), $r = 2$.

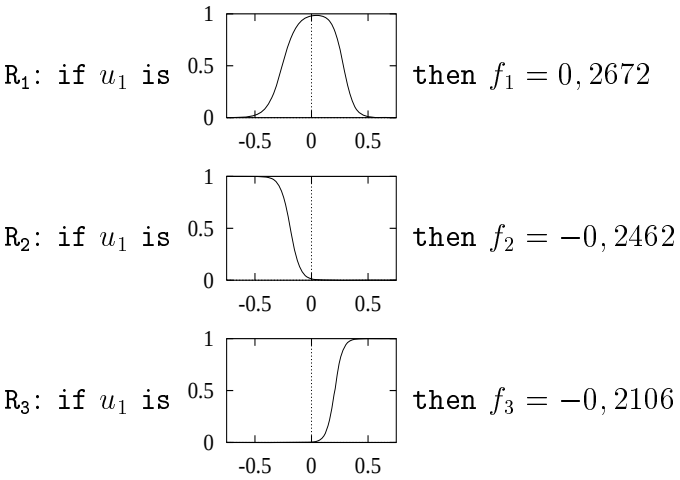

Fig. 12. Model for two Gaussian bell functions, (constant consequences), $r = 3$.

References

[1] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its application to modeling and control", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 15, no. 1, pp. 116–132, 1985.

[2] M. Sugeno and G. Kang, "Structure identification of fuzzy model", *Fuzzy Sets and Systems*, vol. 26, no. 1, pp. 15–33, 1988.

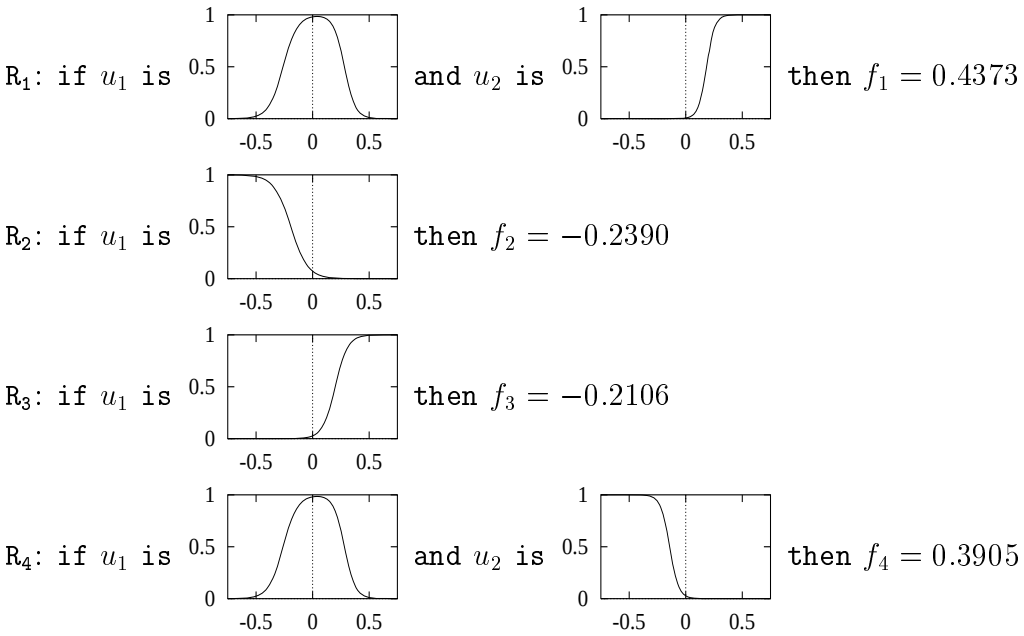

Fig. 13. Model for two Gaussian bell functions, (constant consequences), $r = 4$.

[3] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation: The RPROP algorithm", in *Proceedings of the IEEE Int. Conf. on Neural Networks (ICNN)*, 1993, pp. 586–591.

[4] A. Zell, N. Mache, T. Sommer, et al., *Stuttgart Neural Network Simulator, Users Manual, Version 3.2*, University of Stuttgart, june 1994.

[5] M. Männle, A. Richard, and T. Dörsam, "Identification of rule-based fuzzy models using the RPROP optimization technique", in *Proceedings of Fourth European Congress on Inteligent Techniques and Soft Computing (EUFIT'96)*, Aachen, Germany, September 1996, pp. 587–591.

[6] M. Männle, "Modellierung nichtlinearer Systeme mit unscharfen Regeln", Diplomarbeit, Universität Karlsruhe, Deutsch-Französisches Institut für Automation und Robotik (IAR), Juli 1995.

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

[8] L. Zadeh, "The birth and evolution of fuzzy logic", *International Journal on General Systems*, vol. 17, pp. 95–105, 1973.

[9] L. Zadeh, "The role of fuzzy logic in modeling, identification and control", *Modeling, Identification, and Control*, vol. 15, no. 3, pp. 191–203, 1994.

[10] W. Pedrycz, "Processing in relational structures: Fuzzy relational equations", *Fuzzy Sets and Systems*, vol. 40, pp. 77–106, 1991.

[11] J. Vrba, "Peak-pattern concept and max–min inverse problem in fuzzy control modeling", *Fuzzy Sets and Systems*, vol. 47, no. 1, pp. 1–11, 1992.

[12] J. Chen, J. Lu, and L. Chen, "An on-line identification algorithm for fuzzy systems", *Fuzzy Sets and Systems*, vol. 64, no. 1, pp. 63–72, 1994.

[13] B. Postlethwaite, "Empirical comparison of methods of fuzzy relational identification", *IEEE Proceedings-D*, vol. 138, no. 3, pp. 199–206, 1991.

[14] A. Böhler, "Lösung von Fuzzy-Relationen-Gleichungen", Diplomarbeit, Institut für Prozeßrechentechnik und Robotik, Fakultät für Informatik, Universität Karlsruhe, Oktober 1994.

[15] M. Sugeno and T. Yasukawa, "A fuzzy-logic-based approach to qualitative modeling", *IEEE Transactions on Fuzzy Systems*, vol. 1, no. 1, pp. 7–31, 1993.

[16] Y. Nakamori and M. Ryoke, "Identification of fuzzy prediction models through hyperellipsoidal clustering", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 24, no. 8, pp. 1153–1173, 1994.

[17] R. Yager and D. Filev, "Unified structure and parameter identification of fuzzy models", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, no. 4, pp. 1198–1205, 1993.

[18] M. Tanaka, J. Ye, and T. Tanino, "Identification of nonlinear systems using fuzzy logic and genetic algorithms", *SYSID*, vol. 1, pp. 301–306, 1994.

[19] B. Cormen, C. Leiserson, and R. Rivest, *Introduction to Algorithms*, MIT Press, 1992.

[20] G. Box and G. Jenkins, *Time Series Analysis, Forecasting and Control*, Holden-Day, 6th edition, 1976.