Technique[1]

Manfred Männle,[2]
French-German Institute for
Automation and Robotics (IAR),
Universität Karlsruhe, Germany,
Phone +49 721 608-3910,
Fax +49 721 370455,
maennle@ira.uka.de

Alain Richard,
Centre de Recherche en Automatique
de Nancy (CRAN), CNRS URA 821,
Université Henri Poincaré, Nancy 1, France,
Phone +33 83912069,
Fax +33 83912030,
arichard@cran.u-nancy.fr

Thomas Dörsam,
Institut für Prozeßrechen-
technik und Robotik (IPR),
Universität Karlsruhe, Germany,
Phone +49 721 608-3656,
Fax +49 721 606740,
doersam@ira.uka.de

ABSTRACT: This article discusses a general approach to fuzzy modeling of nonlinear MISO (multiple input, single output) systems. The method of fuzzy modeling is divided in two parts: structure modeling and parameter optimization. Since for this two optimization techniques can be chosen independently, we use a heuristic search for structure modeling and propose the powerful nonlinear optimization algorithm RPROP [Riedmiller 93, Zell 94] for the parameter optimization. We do not transform the fuzzy model into a Neural Network form as usual for many Neuro-Fuzzy approaches but adapt the optimization algorithm to the given fuzzy model. This article is the English publication of the master thesis [Männle 95].

1 INTRODUCTION

In this paper 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 identification or *fuzzy modeling* [Zadeh 94].

Accuracy of prediction and model simplicity are contradictory. The more parameters a model has, the more accurate it can reproduce the input-output relation, but the less it is interpretable by humans. In this paper we want to deal with complex systems with many input variables and a large amount of measured data. For these problems, methods yielding a quadratic or even exponential amount of rules (depending on the number of input variables) are not applicable. To get a still interpretable model we chose fuzzy *rule-based* modeling. The initial model is iteratively enlarged and refined until the desired accuracy or a maximal number of fuzzy rules is reached. Our fuzzy model's structure is similar to the TSK-model's [Zadeh 94] which was first proposed by Takagi, Sugeno, and Kang in [Takagi 85, Sugeno 88]. In the TSK model the rule premises can be considered as an input space partitioning and the rule consequences as local models, valid in the rule's partition. The fuzziness of the premises yield fuzzy transitions between the local models. To identify a TSK-model, the model structure, i.e. the input space partitioning, and the parameters must be optimized.

For TSK-like models, structure and parameter optimization techniques can be chosen independently from each other. So the idea we follow in this paper is to apply a powerful optimization technique for parameter optimization. We chose RPROP [Riedmiller 93, Zell 94], a sophisticated nonlinear optimization algorithm which was originally developed for Neural Network training. RPROP has several advantages: It is robust, i.e. it works well with its standard adjustment. It is a fast algorithm, comparable with the method from Levenberg/Marquardt [Hagan 94]. It is simple to apply, since it only needs the first derivation of the error function. For the application of RPROP we need derivable membership functions. Therefore, we chose sigmoid membership functions who also yield easy to calculate derivations, see (12) and (14). To specify regions in the input universe, bell functions are formed by two overlapping sigmoid curves, which are multiplied.

In the following chapter we describe the used fuzzy model. We adapt RPROP to the given parameter optimization problem and we briefly show the necessary derivations. Then we describe the structure optimization by a heuristic search algorithm. In chapter 3 we test its performance with a benchmark and give an example of modeling a dynamic process. Chapter 4 closes this paper by some conclusions and perspectives.

2 FUZZY IDENTIFICATION USING RPROP

Fuzzy model identification is done by approximating m measured or simulated 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

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

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

[2]M. Männle is now with the Institute of Computer Design and Fault Tolerance, Universität Karlsruhe, Germany.

2.1 The Fuzzy Model

We use sigmoid membership functions given by

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

with the parameters μ and σ to optimize. 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. We consider two possibilities of consequences. The "full" consequence $p_{0k} + p_{1k} \cdot u_1 + \ldots + p_{Nk} \cdot u_N$ defines an arbitrary hyperplane in the input-output space. The "constant" consequence p_{0k}, consisting only of one input variable independent parameter, yields a hyperplane which is orthogonal to the output dimension (y-axis). The form of the used fuzzy rules is

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

for $I = \emptyset$ (inear model). The *fuzzy model* is then given by r fuzzy rules, i.e. $M_R = \{R_1, \ldots, R_r, I_1, \ldots, I_r\}$. Choosing the *product* as t-norm, the membership w_k of $\vec{u}_q$ to the rule R_k is

$$w_k(\vec{u}_q) = \prod_{j \in I_k} F_{jk}(u_{jq}). \tag{4}$$

For $k = 1, \ldots, r$, regarding

$$v_k(\vec{u}) := \frac{w_k(\vec{u})}{\sum_{i=1}^{r} w_i(\vec{u})}, \text{ we get for all } \vec{u}\text{:} \quad \sum_{r=1}^{R} v_r(\vec{u}) = 1, \tag{5}$$

where v_r can be considered as a possibility distribution. The model output is calculated via *product inference* and *weighted average* by

$$\hat{y}(\vec{u}) \quad = \quad \frac{\sum\limits_{k=1}^{r} w_k(\vec{u}) \cdot f_k(\vec{u})}{\sum\limits_{k=1}^{r} w_k(\vec{u})}. \tag{6}$$

2.2 Parameter Identification

Parameter identification is performed by minimizing the prediction error $||\vec{\varepsilon}||^2 := ||\vec{y} - \vec{\hat{y}}\,||^2$, where $\vec{\hat{y}} := (\hat{y}_1, \ldots, \hat{y}_m)^T$ with $\hat{y}_i := \hat{y}(\vec{u}_i)$.

The nonlinear optimization algorithm RPROP [Riedmiller 93, Zell 94] 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. At every iteration t, for all example vectors $\vec{u}_i$, $i = 1, \ldots, m$, the output $\hat{y}(\vec{u}_i)$ and $||\vec{\varepsilon}^{\,t}||^2$ is calculated. Then, regarding the sign change of the partial gradient $\frac{\partial||\vec{\varepsilon}^{\,t}||^2}{\partial p_j}$, the parameter p_j is updated. It is

$$p_j^t = \left\{ \begin{array}{ll} p_j^{t-1} - \Delta_j^t & \text{if } \frac{\partial||\vec{\varepsilon}^{\,t}||^2}{\partial p_j} > 0 \\ p_j^{t-1} + \Delta_j^t & \text{if } \frac{\partial||\vec{\varepsilon}^{\,t}||^2}{\partial p_j} < 0 \\ p_j^{t-1} & \text{else} \end{array} \right. \quad \text{for } \frac{\partial||\vec{\varepsilon}^{\,t-1}||^2}{\partial p_j} \cdot \frac{\partial||\vec{\varepsilon}^{\,t}||^2}{\partial p_j} > 0 \text{ and } p_j^t = p_j^{t-2} \text{ for } \frac{\partial||\vec{\varepsilon}^{\,t-1}||^2}{\partial p_j} \cdot \frac{\partial||\vec{\varepsilon}^{\,t}||^2}{\partial p_j} < 0. \tag{7}$$

Also the step Δ_j^t is updated. A sign change, indicating a too large step, is treated by reducing the step, otherwise the step will be enlarged. Therefore, the step update is

$$\Delta_j^t = \left\{ \begin{array}{ll} \eta^+ \cdot \Delta_j^{t-1} & \text{if } \frac{\partial||\vec{\varepsilon}^{\,t-1}||^2}{\partial p_j} \cdot \frac{\partial||\vec{\varepsilon}^{\,t}||^2}{\partial p_j} > 0 \\ \eta^- \cdot \Delta_j^{t-1} & \text{if } \frac{\partial||\vec{\varepsilon}^{\,t-1}||^2}{\partial p_j} \cdot \frac{\partial||\vec{\varepsilon}^{\,t}||^2}{\partial p_j} < 0 \end{array} \right., \text{ else } \Delta_j^t = \Delta_j^{t-1}, \tag{8}$$

with $0 < \eta^- < 1 < \eta^+$. Practical tests gave best results for $\eta^+ = 1.2$ and $\eta^- = 0.5$ [Riedmiller 93]. At he beginning, all steps are initialized with a $\Delta_0 > 0$. Since the steps are also adapted during the optimization, the algorithm is largely independent from this initialization value. 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 apply RPROP to the given optimization problem, in 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.

The Consequence Parameters:

Using the Euclidean norm we obtain $\|\vec{\varepsilon}\|^2 = \sum_{i=1}^{m} \varepsilon_i^2 = \sum_{i=1}^{m} \left(y_i - \sum_{q=1}^{r} v_q(\vec{u}_i) \cdot f_q(\vec{u}_i) \right)^2$. For the consequence parameters we get with $u_{0i} := 1$ for $k = 1, \ldots, r$ and $\forall j \in I_k$

$$\frac{\partial \|\vec{\varepsilon}^t\|^2}{\partial p_{jk}} = -2 \sum_{i=1}^{m} \varepsilon_i \cdot v_k(\vec{u}_i) \cdot u_{ji}, \quad \text{since} \quad \frac{\partial f_q(\vec{u}_i)}{\partial p_{jk}} = u_{ji} \quad \text{and} \quad \varepsilon_i := y_i - \hat{y}_i. \tag{9}$$

The Premise Parameters:

We obtain the premise parameter's derivations for all $j \in I_k$ and $k = 1, \ldots, r$

$$\frac{\partial \|\vec{\varepsilon}\|^2}{\partial \mu_{jk}} = -2 \sum_{i=1}^{m} \left(\varepsilon_i \cdot \sum_{q=1}^{r} f_q(\vec{u}_i) \cdot \frac{\partial v_q(\vec{u}_i)}{\partial \mu_{jk}} \right) \quad \text{and} \quad \frac{\partial \|\vec{\varepsilon}\|^2}{\partial \sigma_{jk}} = -2 \sum_{i=1}^{m} \left(\varepsilon_i \cdot \sum_{q=1}^{r} f_q(\vec{u}_i) \cdot \frac{\partial v_q(\vec{u}_i)}{\partial \sigma_{jk}} \right). \tag{10}$$

By deriving (5) we get for all $q = 1, \ldots, r; q \neq k$

$$\frac{\partial v_q(\vec{u}_i)}{\partial \mu_{jk}} = \frac{-w_q(\vec{u}_i)}{\left(\sum\limits_{s=1}^{r} w_s(\vec{u}_i) \right)^2} \cdot \frac{\partial w_k(\vec{u}_i)}{\partial \mu_{jk}} \quad \text{and for } q = k\text{:} \quad \frac{\partial v_k(\vec{u}_i)}{\partial \mu_{jk}} = \frac{\left(\sum\limits_{s=1}^{r} w_s(\vec{u}_i) \right) - w_k(\vec{u}_i)}{\left(\sum\limits_{s=1}^{r} w_s(\vec{u}_i) \right)^2} \cdot \frac{\partial w_k(\vec{u}_i)}{\partial \mu_{jk}} \tag{11}$$

with

$$\frac{\partial w_k(\vec{u}_i)}{\partial \mu_{jk}} = \prod_{\substack{d \in I_k \\ d \neq j}} F_{dk}(u_{di}) \cdot \frac{\partial F_{jk}(u_{ji})}{\partial \mu_{jk}} \quad \text{and} \quad \frac{\partial F_{jk}(u_{ji})}{\partial \mu_{jk}} = \frac{-e^{\sigma_{jk} \cdot (u_{ji} - \mu_{jk})} \cdot (-\sigma_{jk})}{\left(1 + e^{\sigma_{jk} \cdot (u_{ji} - \mu_{jk})}\right)^2} \tag{12}$$

$$= \sigma_{jk} F_{jk}(u_{ji}) \left(1 - F_{jk}(u_{ji})\right).$$

Correspondingly we get for all $q = 1, \ldots, r; q \neq k$

$$\frac{\partial v_q(\vec{u}_i)}{\partial \sigma_{jk}} = \frac{-w_q(\vec{u}_i)}{\left(\sum\limits_{s=1}^{r} w_s(\vec{u}_i) \right)^2} \cdot \frac{\partial w_k(\vec{u}_i)}{\partial \sigma_{jk}} \quad \text{and for } q = k\text{:} \quad \frac{\partial v_k(\vec{u}_i)}{\partial \sigma_{jk}} = \frac{\left(\sum\limits_{s=1}^{r} w_s(\vec{u}_i) \right) - w_k(\vec{u}_i)}{\left(\sum\limits_{s=1}^{r} w_s(\vec{u}_i) \right)^2} \cdot \frac{\partial w_k(\vec{u}_i)}{\partial \sigma_{jk}} \tag{13}$$

with

$$\frac{\partial w_k(\vec{u}_i)}{\partial \sigma_{jk}} = \prod_{\substack{d \in I_k \\ d \neq j}} F_{dk}(u_{di}) \cdot \frac{\partial F_{jk}(u_{ji})}{\partial \sigma_{jk}} \quad \text{and} \quad \frac{\partial F_{jk}(u_{ji})}{\partial \sigma_{jk}} = \frac{-e^{\sigma_{jk} \cdot (u_{ji} - \mu_{jk})} \cdot (u_{ji} - \mu_{jk})}{\left(1 + e^{\sigma_{jk} \cdot (u_{ji} - \mu_{jk})}\right)^2} \tag{14}$$

$$= (\mu_{jk} - u_{ji}) F_{jk}(u_{ji}) \left(1 - F_{jk}(u_{ji})\right).$$

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 used the heuristic search described in [Takagi 85, Sugeno 88]. 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. Starting from scratch, the algorithm needs

$$T(r_{max}, N, m, I) = \mathrm{O}\left(\sum_{r=1}^{r_{max}} r \cdot N \cdot (rNmI) \right) = \mathrm{O}\left(r_{max}^3 N^2 m I \right) \tag{15}$$

calculation steps in the worst case. In (15), I is the maximal number of RPROP iterations. In practical applications we found $I = 100$ to be sufficient as a maximum border. We use the R^2 criterion to calculate the model quality. It is defined by

$$R^2 = 1 - \frac{V(\varepsilon)}{V(y)} = 1 - \frac{\sum_{i=1}^{n} (\varepsilon_i - E(\varepsilon_i))^2}{\sum_{i=1}^{n} (y_i - E(y_i))^2} \tag{16}$$

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 [Takagi 85]. Then, the RPROP algorithm is performed on A whereas 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 and the training procedure is interrupted.

3 EXAMPLES

3.1 Frank's Benchmark

To test the algorithm's performance we chose a benchmark from I. Frank [Frank 95]. It contains four four-dimensional static functions (A–D) and an example based on measured data from food industry (E).

$$\text{A:} \quad y = u_1 + u_2 + u_3 + u_4 \tag{17}$$
$$\text{B:} \quad y = 2(u_1 - 1)^2 + 2(u_2 - 1)^2 + \sin(2\pi u_3) + \ln(u_4 + 0.1) \tag{18}$$
$$\text{C:} \quad y = \sin(u_1 + u_2 + u_3 + u_4) + \ln(u_1 + u_2 + u_3 + u_4) \tag{19}$$
$$\text{D:} \quad y = 10\sin(u_1 u_2) + 20(u_3 - 0.5)^2 + 5u_4 \tag{20}$$

Data of E are found in [Frank 95]. Each model was examined with four levels of white noise which was added with signal-to-noise ratios SNR = 1, 3, 7, and ∞, respectively. This corresponds to maximal reachable R^2 values of 0.5, 0.9, 0.98, and 1.0. Besides that, each models was also examined with four additional input variables which only contain white noise. So, there are 5x4x2 = 40 models to be examined. For all these 40 models, training sets (100 points) and test sets (1000 points) were generated by choosing the x-variables uniformly in $[0, 1]$. Data generation, modeling, and test was performed 5 times and a final average R^2 was calculated for each model A–E. For these models we used fuzzy rules containing full consequences. Table 1 shows the overall result, the sum of all R^2s of all forty models for the developed fuzzy algorithm and the four algorithms from [Frank 95]. A value of $10 \cdot (1.0 + 0.98 + 0.9 + 0.5) = 33.8$ means an optimal identification of all models.

Algorithm:	FUZZY	MARS	ACE	SMART	CART
$\sum R^2$:	28.5	27.8	24.2	22.9	16.3

Table 1: Comparison of five modeling algorithms.

3.2 Box/Jenkins' Gas Furnace

We investigated the modeling of a dynamic process using the well-known example given by Box and Jenkins. The process is a gas furnace with a single input $u(t)$ (gas flow rate) and a single output $y(t)$ (CO_2-concentration). The measured data are found in [Box 76]. All 296 data were normalized. The first 145 points were used for the modeling algorithm. Since this is a dynamic process and we have no prior knowledge, we chose $u_{t-1}, \ldots, u_{t-6}$ and $y_{t-1}, \ldots, y_{t-4}$ as input variables. For better interpretability we used fuzzy rules with constant consequences, despite they are less powerful.

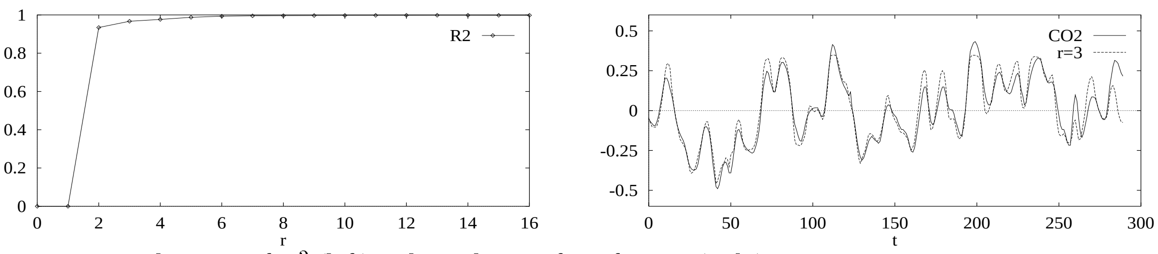

Figure 1: Development of R^2 (left) and simulation of gas furnace (right), $r = 3$.

The development of R^2 is shown in figure 1 (left). With $r = 3$ we have good, with $r = 7$ nearly perfect estimations on the training data. Now let us test the algorithm's performance in the *simulation* where the model output is given by the recursive function $\hat{y}_t = \hat{y}(u_{t-1}, \ldots, u_{t-6}, \hat{y}_{t-1}, \ldots, \hat{y}_{t-4})$. Even the simple three-rule model shows a good simulation on all 296 data points (figure 1, right). Note, that models containing more rules or trained on all 296 data points show better results, and especially models with full consequences achieve a better accuracy. What we wanted to show here is that even a very simple and easily interpretable model with three rules and constant consequences can already give satisfying simulation results. The identified model is listed in figure 2, its linguistic interpretation in figure 3.

4 CONCLUSION

In this paper we present an approach of rule-based fuzzy modeling. The initial fuzzy model is iteratively refined and parameters are adapted at each refining step until a desired accuracy or a maximal allowed model complexity is reached. We propose the use of RPROP for parameter optimization. The developed method is examined in several examples.

The algorithm shows good performance in comparison with other nonlinear identification techniques. At the regarded benchmark it is the best at model accuracy. Already simple models can show satisfying precision as it is shown by the simulation of a gas furnace as example for a dynamic process.

In (15) 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 the exclusion of input variables which are not involved in premises during several epochs.

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*. Regarding *single* input-output pairs can cause a too big influence of possible outliers.

A more sophisticated technique to find the model structure could also yield more accurate results. Possible techniques are meta algorithms like for example genetic algorithms [Tanaka 94]. All conditions for the application of a similar 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.

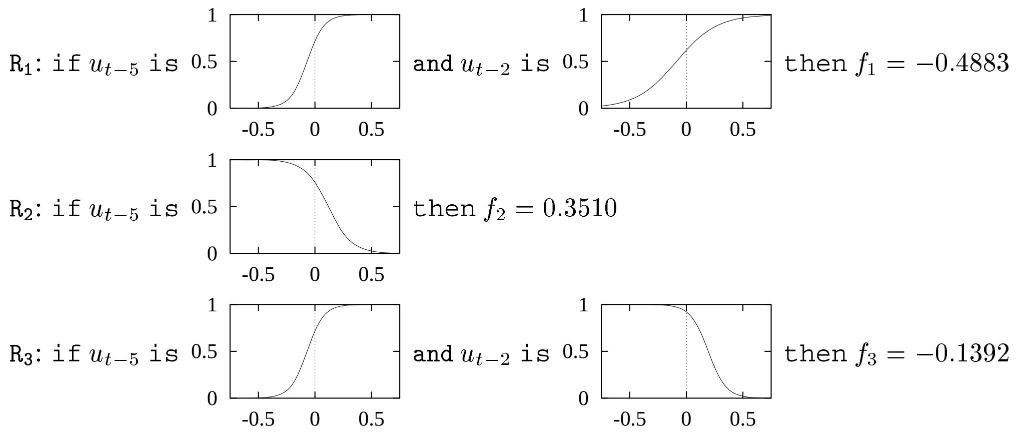

Figure 2: Model for Box/Jenkins' gas furnace, (constant consequences), $r = 3$.

$\mathtt{R}_1$: if u_{t-5} is BIG^1 and u_{t-2} is BIG^2 then $\mathrm{CO}_2{-}\mathrm{rate}$ is $\mathrm{SMALL}^{\mathrm{out}}$
$\mathtt{R}_2$: if u_{t-5} is SMALL^1 then $\mathrm{CO}_2{-}\mathrm{rate}$ is $\mathrm{BIG}^{\mathrm{out}}$
$\mathtt{R}_3$: if u_{t-5} is BIG^1 and u_{t-2} is SMALL^2 then $\mathrm{CO}_2{-}\mathrm{rate}$ is $\mathrm{MEDIUM}^{\mathrm{out}}$

Figure 3: Linguistic interpretation of three-rule model for Box/Jenkins' gas furnace (figure 2).

REFERENCES

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

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.

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

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.

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.

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.