A Neurofuzzy Training Method for Mamdani-Like Structures

L. SCHNITMAN[1] , J.A.M. FELIPPE DE SOUZA[2] , T. YONEYAMA[1]
[1]Instituto Tecnológico de Aeronáutica, 12.228-900 S.J. dos Campos, SP – Brazil
[2]Universidade da Beira Interior, 6201-001 Covilhã – Portugal
leizer@ele.ita.cta.br ; felippe@demnet.ubi.pt ; takashi@ele.ita.cta.br

Abstract: - This paper proposes a backpropagation method to train the Mamdani-like fuzzy structures. The idea is to train the proposed fuzzy structure maintaining a smaller computational effort as well as the Takagi-Sugeno-Kang (TSK) case.

Key-Words: - Fuzzy systems, fuzzy sets, discrete fuzzy

1 Introduction

The Mamdani fuzzy structure as proposed by E.H. Mamdani in [21] reflects the ideas that are originally proposed by L.A.Zadeh [37] to providing an implementation that is intuitive. Because of the heuristics insight, it has received widespread acceptance in industrial and academic media and it is well suited in terms of interactions with humans. In spite of these qualities, the computational effort that is required in the Mamdani *defuzzification* procedure may be prohibitive and it may become impracticable to implement in real cases, especially when learning/training are necessary.

The *gradient descent* method is the most usual technique to train neural networks and neurofuzzy structures but it cannot be applied directly to the original fuzzy system because the inference process is represented by functions such as *min* and *max* which are not differentiable. In the literature two basic approach are proposed: replace these functions, using differentiable ones or do not use specifically the gradient but other learning procedures. Basing on this first idea, Berenji and Khedkar [4] propose the GARIC (Generalized ARIC - Approximate Reasoning-based Intelligent Control) that use the *softmin* operator and Jang [13] proposes the ANFIS (Adaptive Network-based Fuzzy Inference System) that use the *sum-product* approach. The basic difficulty with these and other approaches is that they become very hard to apply to Mamdani fuzzy structures and they are usually restricted to the TSK [32], [34] cases (e.g. [1], [2], [5], [8], [9], [11], [12], [14], [15], [16], [18], [20], [22], [23], [26], [31], [33], [35], [36]. Another usual approach is based on the *fuzzy neuron* (e.g. [6], [17], [24], and [25]) that uses the neural network structure but generalize the artificial neuron concepts to get the fuzzy reasoning.

This article is based on a *Mamdani-like* structure that was previously proposed by the authors in [29] and describes the training procedures analogous to the backpropagation used with neural nets. It shows that the computational effort becomes compatible with TSK, but keeping the fuzzy reasoning yielding consequent membership functions (MF), thus rescuing the fuzzy concepts as proposed by Mamdani, specially referring to the consequent MF properties.

Section 2 contains a brief review of the *Mamdani-like* structure proposed by the authors and its mathematical description. Section 3 describes the training procedures and related expressions. In section 4 some numerical results are provided and the conclusions are presented in section 5.

2 Mamdani-like Propositions

One of the basic differences between the Mamdani and TSK fuzzy models lies on the fact that the consequences are, respectively, fuzzy and crisp sets. Hence, the computation procedure required to obtain the output signals are distinct. While in the case of TSK fuzzy models the output is computed with a very simple formula (*weighted average, weighted sum*), Mamdani fuzzy models require higher computational effort in the *defuzzification* procedure. Therefore, in spite of the Mamdani fuzzy characteristics, the TSK fuzzy structures becomes the most common way to apply fuzzy reasoning, specially when the fuzzy models requires training and optimization of membership functions.

In [27] and [28] the authors apply the backpropagation directly to the discrete Mamdani structure, using the *sum-product* composition to derive the gradient expressions. In spite of the successful results, the computational effort is considerable. In [29] the authors propose a new *Mamdani-like* structure that simply computes the centroids of each consequent MF before aggregate them. Some authors refers to [19] (pp. 386) and [14] (pp. 80) that tries to show that *defuzzifying* each rule individually generates the same result that is obtained when the aggregated rules are *defuzzified*. In [29] the authors show that this is only true for a very specifical case and not valid in the general case. Initially a brief review of the author's *Mamdani-like* structure is presented.

For sake of simplicity, avoiding cumbersome notation, the figure 1 represents an example for the SISO case. For the general case the equations are similar, but it would require notational adaptations.

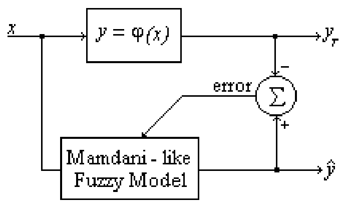

Figure 1: SISO training block diagram

2.1 A fuzzy rule base

As proposed in the figure 1, define x as the input and $\hat{y}$ as the fuzzy output signal. Then consider the antecedent/consequent MF and the rule-base:

a) $X_1,...,X_n$ as the n antecedent MF
b) $Y_1,...,Y_m$ as the m consequent MF
c) $R_1,...,R_r$ as the r rules that compose the rulebase

Generally the fuzzy reasoning can be expressed through rules of form:

$$\text{If } x \text{ is } X_i \text{ then } y \text{ is } Y_i \tag{1}$$

where $X_i \in \{X_1 ,..., X_n\}$ and $Y_i \in \{Y_1 ,..., Y_m\}$

In order to apply the gradient descent procedure the inference operators should be differentiable. This paper uses the *sum-product* composition as usual in practical implementation (e.g. [3], [5], [7], [10], [11], [14], [30], and [31]). The operators satisfy the differentiability property and will be useful to achieve the desired results.

2.2 *Defuzzification*

Define the function *Dfz(MF)* to be used to *defuzzifying* a given MF (e.g. triangular, trapezoidal, gaussian,...) based on its own parameters and using the centroid method, that means:

1. For triangular MFs, $Y_i=[a \;\; b \;\; c]$ where a,b,c are the breakpoint parameters and h is its height. The *defuzzification* formula becomes:

$$Dfz(Y_i)=\frac{h}{3}(a+b+c) \tag{2}$$

2. For trapezoidal MFs, $Y_i=[a \;\; b \;\; c \; d\,]$ where a,b,c,d are the breakpoint parameters and h is its height. The *defuzzification* formula becomes:

$$Dfz(Y_i)=\frac{h}{3}\left(\frac{d^2+c^2-b^2-a^2+c.d-a.b}{d+c-b-a}\right) \tag{3}$$

3. For gaussian form, $Y_i= [\sigma, \; c]$ where σ,c are the parameters, similar to $N(c,\sigma^2)$, and h is its height. The *defuzzification* formula becomes:

$$Dfz(Y_i)=h.c \tag{4}$$

These equations only refer to some simple MF just because these expressions are simples. In general cases, they must be extended to obtain the fuzzy output.

2.3 Computation of output

For each rule i, define T as the vector of each consequent MF centroid, represented as T_i.

$$T_i=Dfz(Y_i) \;, \quad \forall i \in [1,r] \tag{5}$$

and let C_i be the consequence of the rule i

$$C_i=\mu_{Xi}(x).T_i \;, \quad \forall i \in [1,r] \tag{6}$$

Then compose the C matrix as the matrix of the rule consequences, that becomes:

$$C=\begin{bmatrix} C_1=\mu_{X^1}(x).T_1 \\ C_2=\mu_{X^2}(x).T_2 \\ \vdots \\ C_r=\mu_{X^r}(x).T_r \end{bmatrix}_{r\times 1} \tag{7}$$

Finally, simply use the *sum* operator to aggregate the rules consequences and obtain the fuzzy output:

$$\hat{y}=\sum_{i=1}^{r} C_i \tag{8}$$

This is graphically represented in figure 2.

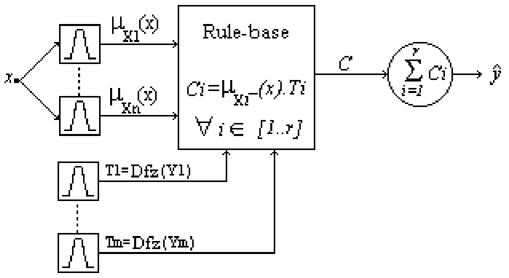

Figure 2: Mamdani-like fuzzy structure

See [29] for details.

3 The Training Algorithm

Consider the trapezoidal MF parameters as shown in figure 3.

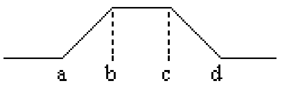

Figure 3: Trapezoidal MF

To train the fuzzy structure, admit that the data pair (x,y_t) is available. Consider y_t as the desired (target) output when the input is x and $e = \hat{y} - y_t$ as the output error. Thus define the pointwise cost function as

$$J(x) = \frac{1}{2}e^2(x) = \frac{1}{2}\left[\hat{y}(x) - y_t(x)\right]^2 \tag{9}$$

Define the input universe of discourse (UD) and set the global cost function as

$$SSE(x) = \frac{1}{2}\sum_{\forall x \in UD}\left[\hat{y}(x) - y_t(x)\right]^2 \tag{10}$$

Define P as the either antecedent or consequent MF matrix of parameters, then the backpropagation training algorithm must compute:

$$P_{k+1} = P_k - \eta_P . \frac{\partial J}{\partial P} \tag{11}$$

where P_{k+1} is the new value of the matrix P based on its last estimate P_k and η_P as the learning rate.
For the *Mamdani-like* fuzzy structure as proposed in figure 2, $\frac{\partial J}{\partial P}$ becomes:

$$\frac{\partial J}{\partial P} = \frac{\partial J}{\partial \hat{y}} . \frac{\partial \hat{y}}{\partial P} \tag{12}$$

Where the required terms are:

$$a)\ \frac{\partial J}{\partial \hat{y}} = e \tag{13}$$

$$b)\ \frac{\partial \hat{y}}{\partial P} = \frac{\partial\left(\sum_{i=1}^{r} C_i\right)}{\partial P} \tag{14}$$

$$c)\ \frac{\partial C_i}{\partial P} = \frac{\partial\left[\mu_{Xi}(x).Dfx(Yi)\right]}{\partial P}, \quad i = 1..r \tag{15}$$

Finally, note that $\mu_{Xi}(x)$ and $Dfz(Y_i)$ only depends on each antecedent/consequent MF parameters respectively then the differentiation becomes trivial and only differ for each kind of MF.

3.1 Practical computation of the gradient

The idea of using *gradient descent* is simple, but its computation requires some care, especially w.r.t. the Jacobian as shown in the sequel.
Define P^A as the antecedent MF matrix of parameters with dimension [4 x n]. That means:

$$P^A = \begin{bmatrix} P^A_{1,a} & P^A_{2,a} & \cdots & P^A_{n,a} \\ P^A_{1,b} & P^A_{2,b} & \cdots & P^A_{n,b} \\ P^A_{1,c} & P^A_{2,c} & \cdots & P^A_{n,c} \\ P^A_{1,d} & P^A_{2,d} & \cdots & P^A_{n,d} \end{bmatrix} \tag{16}$$

where $P^A_{v,w}$ is the w parameter, $w \in \{a,b,c,d\}$, of the v th antecedent MF, $v \in [1,n]$. The $\frac{\partial \hat{y}}{\partial P}$ expression must be computed for the gradient w.r.t. each parameter. Then the equation 11 becomes:

$$\left[P^A_{k+1}\right]_{4xn} = \left[P^A_k\right]_{4xn} - \eta_A .e. \left[\frac{\partial J}{\partial P^A}\right]_{4xn} \tag{17}$$

And refer to equation 8 to obtain:

$$\frac{\partial \hat{y}}{\partial P^A} = \begin{bmatrix} \frac{\partial\left(\sum_{i=1}^{r} C_i\right)}{P^A_{1,a}} & \frac{\partial\left(\sum_{i=1}^{r} C_i\right)}{P^A_{2,a}} & \cdots & \frac{\partial\left(\sum_{i=1}^{r} C_i\right)}{P^A_{n,a}} \\ \frac{\partial\left(\sum_{i=1}^{r} C_i\right)}{P^A_{1,b}} & \frac{\partial\left(\sum_{i=1}^{r} C_i\right)}{P^A_{2,b}} & \cdots & \frac{\partial\left(\sum_{i=1}^{r} C_i\right)}{P^A_{n,b}} \\ \frac{\partial\left(\sum_{i=1}^{r} C_i\right)}{P^A_{1,c}} & \frac{\partial\left(\sum_{i=1}^{r} C_i\right)}{P^A_{2,c}} & \cdots & \frac{\partial\left(\sum_{i=1}^{r} C_i\right)}{P^A_{n,c}} \\ \frac{\partial\left(\sum_{i=1}^{r} C_i\right)}{P^A_{1,d}} & \frac{\partial\left(\sum_{i=1}^{r} C_i\right)}{P^A_{2,d}} & \cdots & \frac{\partial\left(\sum_{i=1}^{r} C_i\right)}{P^A_{n,d}} \end{bmatrix}_{4xn} \tag{18}$$

but

$$C_i = \mu_{Xi}(x).Dfz(Y_i)\ , \quad \forall i \in [1,r] \tag{19}$$

and

$$\sum_{i=1}^{r} C_i = \mu_{X1}(x).Dfz(Y_1) + \ldots + \mu_{Xr}(x).Dfz(Y_r) \tag{20}$$

then

$$\frac{\partial\left(\sum_{i=1}^{r} C_i\right)}{\partial P^A_{v,w}} = \frac{\partial\left[\mu_{X1}(x).Dfz(Y_1)\right]}{\partial P^A_{v,w}} + \ldots + \frac{\partial\left[\mu_{Xr}(x).Dfz(Y_r)\right]}{\partial P^A_{v,w}} \tag{21}$$

As the consequent MF does not depend on the antecedent MF parameters, the equation 21 becomes:

$$\frac{\partial\left(\sum_{i=1}^{r} C_i\right)}{\partial P^A_{v,w}} = Dfz(Y_1).\frac{\partial\left[\mu_{X1}(x)\right]}{\partial P^A_{v,w}} + \ldots + Dfz(Y_r).\frac{\partial\left[\mu_{Xr}(x)\right]}{\partial P^A_{v,w}} \tag{22}$$

Analogously one can obtain the consequent MF training expressions. Define P^C as the consequent

MF matrix of parameters with dimension [4 x m]. That is,

$$P^C = \begin{bmatrix} P^C_{1,a} & P^C_{2,a} & \cdots & P^C_{n,a} \\ P^C_{1,b} & P^C_{2,b} & \cdots & P^C_{n,b} \\ P^C_{1,c} & P^C_{2,c} & \cdots & P^C_{n,c} \\ P^C_{1,d} & P^C_{2,d} & \cdots & P^C_{n,d} \end{bmatrix} \tag{23}$$

where $P^A_{v,w}$ is the w parameter, $w \in \{a,b,c,d\}$, of the vth consequent MF, $v \in [1,m]$. The $\frac{\partial \hat{y}}{\partial P}$ expression must be computed for the gradient w.r.t each parameter. Then the equation 11 becomes:

$$\left[P^C_{k+1}\right]_{4xm} = \left[P^C_{k}\right]_{4xm} - \eta_C . e. \left[\frac{\partial J}{\partial P^C}\right]_{4xm} \tag{24}$$

And referring to equation 8 one obtains that:

$$\frac{\partial \hat{y}}{\partial P^C} = \begin{bmatrix} \frac{\partial\left(\sum_{i=1}^{r} C_i\right)}{P^C_{1,a}} & \frac{\partial\left(\sum_{i=1}^{r} C_i\right)}{P^C_{2,a}} & \cdots & \frac{\partial\left(\sum_{i=1}^{r} C_i\right)}{P^C_{n,a}} \\ \frac{\partial\left(\sum_{i=1}^{r} C_i\right)}{P^C_{1,b}} & \frac{\partial\left(\sum_{i=1}^{r} C_i\right)}{P^C_{2,b}} & \cdots & \frac{\partial\left(\sum_{i=1}^{r} C_i\right)}{P^C_{n,b}} \\ \frac{\partial\left(\sum_{i=1}^{r} C_i\right)}{P^C_{1,c}} & \frac{\partial\left(\sum_{i=1}^{r} C_i\right)}{P^C_{2,c}} & \cdots & \frac{\partial\left(\sum_{i=1}^{r} C_i\right)}{P^C_{n,c}} \\ \frac{\partial\left(\sum_{i=1}^{r} C_i\right)}{P^C_{1,d}} & \frac{\partial\left(\sum_{i=1}^{r} C_i\right)}{P^C_{2,d}} & \cdots & \frac{\partial\left(\sum_{i=1}^{r} C_i\right)}{P^C_{n,d}} \end{bmatrix}_{4xm} \tag{25}$$

and then

$$\frac{\partial\left(\sum_{i=1}^{r} C_i\right)}{\partial P^C_{v,w}} = \frac{\partial[\mu_{X1}(x).Dfz(Y_1)]}{\partial P^C_{v,w}} + \ldots + \frac{\partial[\mu_{Xr}(x).Dfz(Y_r)]}{\partial P^C_{v,w}} \tag{26}$$

As the antecedent MF does not depend on the consequent MF parameters, the equation 26 becomes:

$$\frac{\partial\left(\sum_{i=1}^{r} C_i\right)}{\partial P^C_{v,w}} = \mu_{X1}(x)\frac{\partial[Dfz(Y_1)]}{\partial P^C_{v,w}} + \ldots + \mu_{Xr}(x)\frac{\partial[Dfz(Y_r)]}{\partial P^C_{v,w}} \tag{27}$$

3.2 Gradient expressions

Note that $\frac{\partial J}{\partial \hat{y}} = e$ is easily obtained but $\frac{\partial \hat{y}}{\partial P}$ requires the $\frac{\partial \mu_{Xi}(x)}{\partial P^A}$ and $\frac{\partial Dfz(Y_i)}{\partial P^A}$ expressions. For instance, using trapezoidal MF as shown in figure 3, it can be expressed as function of its own parameters by:

$$\mu(x) = \begin{cases} 0 & if\ x \le a \\ \frac{x-a}{b-a} & if\ a < x < b \\ 1 & if\ b \le x \le c \\ \frac{x-d}{c-d} & if\ c < x < d \\ 0 & if\ x \ge d \end{cases} \tag{28}$$

and the gradient of the MF with respect to those parameters {a,b,c,d} are given by:

$$\frac{\partial \mu(x)}{\partial a} = \begin{cases} \frac{\mu(x) - 1}{b-a} & if\ a < x < b \\ 0 \quad otherwise \end{cases}$$

$$\frac{\partial \mu(x)}{\partial b} = \begin{cases} \frac{-\mu(x)}{b-a} & if\ a < x < b \\ 0 \quad otherwise \end{cases} \tag{29}$$

$$\frac{\partial \mu(x)}{\partial c} = \begin{cases} \frac{-\mu(x)}{c-d} & if\ c < x < d \\ 0 \quad otherwise \end{cases}$$

$$\frac{\partial \mu(x)}{\partial d} = \begin{cases} \frac{\mu(x) - 1}{c-d} & if\ c < x < d \\ 0 \quad otherwise \end{cases}$$

Those are necessary to update the antecedent MF. It is important to note that the gradients are undefined at points $x=a$, $x=b$, $x=c$ and $x=d$. However, this fact does not yield any difficulty in the present formulation.
To update the consequent ones, derive the equation 3 and obtain the $\frac{\partial Dfz(Y_i)}{\partial P^C}$ expressions:

$$\frac{\partial Dfz(Y_i)}{\partial a} = \frac{h}{3}\left[\frac{-(2a+b).K_1 + K_2}{K_1^2}\right]$$

$$\frac{\partial Dfz(Y_i)}{\partial b} = \frac{h}{3}\left[\frac{-(2b+a).K_1 + K_2}{K_1^2}\right] \tag{30}$$

$$\frac{\partial Dfz(Y_i)}{\partial c} = \frac{h}{3}\left[\frac{(2c+d).K_1 - K_2}{K_1^2}\right]$$

$$\frac{\partial Dfz(Y_i)}{\partial d} = \frac{h}{3}\left[\frac{(2d+c).K_1 + K_2}{K_1^2}\right]$$

where

$$K_1 = d + c - b - a \tag{31}$$

and

$$K_2 = d^2 + c^2 - b^2 - a^2 + cd - ab \tag{32}$$

3.3 Computation of the Jacobian

The Jacobian defined in 18 and 25 are easy to compute. Note that if a MF is not in a rule, the gradient result is zero. Thus, it is possible to reduce the computation effort just computing the gradient for the MFs that are included in each rule. To compute the Jacobian, use the following algorithm:

Algorithm 1: Jacobian computation

```
a) Define $D^A = \frac{\partial \hat{y}}{\partial P^A}$ and $D^C = \frac{\partial \hat{y}}{\partial P^C}$
b) Initialize $D^A$ = zeros(4,$n$)
c) Initialize $D^C$ = zeros(4,$m$)
d) Read the input signal $x$
e) For each rule $i = 1..r$
    Compute $Dfz(Yi)$
    Compute $\mu_{Xi}(x)$
    For each antecedent MF $v = 1..n$
        If the MF$_v$ is included in the rule $i$
            • $D^A(1,v) = D^A(1,v) + Dfz(Y_i).\ \frac{\partial \mu_{Xi}(x)}{\partial P^A_{v,a}}$
            • $D^A(2,v) = D^A(2,v) + Dfz(Y_i).\ \frac{\partial \mu_{Xi}(x)}{\partial P^A_{v,b}}$
            • $D^A(3,v) = D^A(3,v) + Dfz(Y_i).\ \frac{\partial \mu_{Xi}(x)}{\partial P^A_{v,c}}$
            • $D^A(4,v) = D^A(4,v) + Dfz(Y_i).\ \frac{\partial \mu_{Xi}(x)}{\partial P^A_{v,d}}$
        End If
    End For
    For each consequent MF $v = 1..m$
        If the MF$_v$ is included in the rule $i$
            • $D^C(1,v) = D^C(1,v) + \mu_{Xi}(x)\frac{\partial[Dfz(Y_i)]}{\partial P^C_{v,a}}$
            • $D^C(2,v) = D^C(2,v) + \mu_{Xi}(x)\frac{\partial[Dfz(Y_i)]}{\partial P^C_{v,b}}$
            • $D^C(3,v) = D^C(3,v) + \mu_{Xi}(x)\frac{\partial[Dfz(Y_i)]}{\partial P^C_{v,c}}$
            • $D^C(4,v) = D^C(4,v) + \mu_{Xi}(x)\frac{\partial[Dfz(Y_i)]}{\partial P^C_{v,d}}$
        End If
    End For
```

3.4 Training membership functions

As proposed in the ANFIS approach, consider the antecedent MF as constants to train the consequents and vice-versa. Consider P^A and P^C as proposed before and use the data pairs (x,y_t) to train the model. The training algorithm becomes:

a) Read x and y_t

b) Propagate the x input through the fuzzy net

Compute $C_i=\mu_{Xi}(x).Dfz(Y_i)\ ,\quad \forall i \in [1,r]$ (33)

Obtain the fuzzy output

$$\hat{y} = \sum_{i=1}^{r} C_i \tag{34}$$

c) Compute the error

$$e = \hat{y} - y_t \tag{35}$$

and

$$\frac{\partial J}{\partial \hat{y}} = e \tag{36}$$

d) Use the *Algorithm* 1 to compute the Jacobian

$$D^A = \left[\frac{\partial \hat{y}}{\partial P^A}\right]_{4xn} \text{ and } D^C = \left[\frac{\partial \hat{y}}{\partial P^C}\right]_{4xm} \tag{37}$$

e) Update the antecedent MF

Consider P^C as constant

Update P^A

$$\left[P^A_{k+1}\right]_{4xn} = \left[P^A_k\right]_{4xn} - \eta_A.e.\left[\frac{\partial J}{\partial P^A}\right]_{4xn} \tag{38}$$

f) Update the consequent MF

Consider P^A as constant

Update P^C

$$\left[P^C_{k+1}\right]_{4xm} = \left[P^C_k\right]_{4xm} - \eta_C.e.\left[\frac{\partial J}{\partial P^C}\right]_{4xm} \tag{39}$$

4 Numerical Results

Let the desired function be:

$$y = x^4 - 3x^3 - 4x^2 + x + 40 \tag{40}$$

and consider the input UD as [-3,5]. The output UD can be [0,200]. Initialize the antecedent and consequent MF as proposed in [28], and for this simple example, 8/10 antecedent/consequent MF are used respectively. The initial *SSE* was 1331 and the figure 4 shows the results after 1000 iterations and the MF parameters becomes:

$$\left[P^A\right] = \begin{bmatrix} -3,00 & -3,00 & -3,00 & -2,00 \\ -3,00 & -2,30 & -2,30 & -1,52 \\ -2,29 & -1,42 & -1,42 & -0,38 \\ -1,30 & 0,55 & 0,55 & 3,24 \\ -0,06 & 2,69 & 2,94 & 4,09 \\ 2,69 & 3,59 & 3,60 & 4,05 \\ 3,59 & 4,28 & 4,30 & 4,87 \\ 4,30 & 5,00 & 5,00 & 5,00 \end{bmatrix}_{8x4} \tag{41}$$

and

$$
\left[P^C\right]=\begin{bmatrix}
-13{,}95 & -1{,}42 & -1{,}39 & 8{,}30 \\
10{,}22 & 22{,}22 & 22{,}29 & 22{,}29 \\
30{,}00 & 44{,}37 & 50{,}10 & 50{,}10 \\
55{,}91 & 66{,}95 & 75{,}84 & 83{,}81 \\
75{,}38 & 88{,}89 & 88{,}89 & 88{,}89 \\
100{,}0 & 111{,}1 & 111{,}1 & 122{,}2 \\
122{,}2 & 133{,}3 & 133{,}3 & 144{,}4 \\
143{,}9 & 155{,}3 & 164{,}7 & 164{,}7 \\
167{,}8 & 177{,}8 & 177{,}8 & 188{,}9 \\
188{,}8 & 200{,}0 & 200{,}1 & 200{,}1
\end{bmatrix}_{10x4} \tag{42}
$$

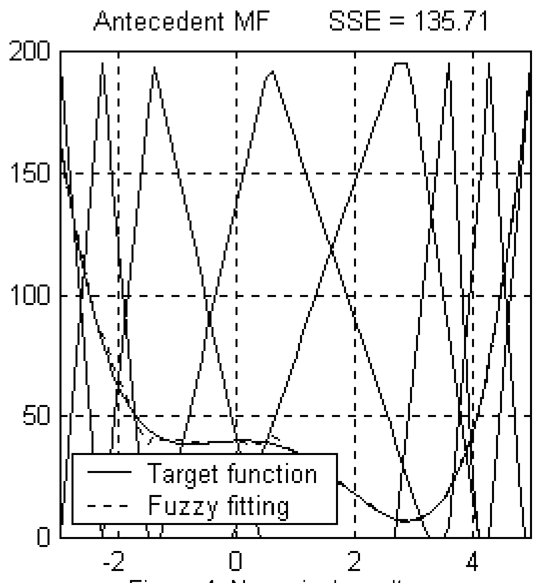

Figure 4: Numerical results

We also compared the computational effort, with the [28] method that uses the discretized Mamdani approach to compute the fuzzy output. It shows that this new approach was at least 15 x faster.

5 Conclusions

The computational effort required in the Mamdani *defuzzification* procedure make it nonpractical, and many fuzzy applications was developed using TSK proposition even losing some essential information contained in the consequent MF. For example, ANFIS is one of the most usual neurofuzzy approach. It becomes very practical because it uses a simple but efficient fuzzy equations (TSK structure) and training procedures, such as gradient descent or least squares.

This article presents a neurofuzzy method to train the antecedent and consequent MF for the Mamdani-like fuzzy structure proposed in [29]. As expected, the computational effort is reduced and becomes compatible with the TSK. The results may remind the TSK approach using the *weighted sum* to compute $\hat{y}$ and consequent MF as constants (zero order TSK), however, the fuzzy properties of the consequent MF are different (See [29] for details).

The advantages of keeping the consequent MF and the lower computational effort to train them, must characterize a very useful tool in application of the concepts of fuzzy reasoning and neurofuzzy techniques such as backpropagation.

Acknowledgement: The authors are indebted to the FAPESP – Fundação de Amparo à Pesquisa do Estado de São Paulo for support under grant 98/16074-1.

References:

[1] M.Akar and U.Ozguner, Stability and Stabilization of Takagi-Sugeno Fuzzy Systems, *Proc. of the 38° IEEE Conf. on Decision and Control*, 1999, pp. 4840-4845.
[2] S.Barada and H.Singh, Generating Optimal Adaptive Fuzzy-Neural Models of Dynamical Systems with Applications to Control, *IEEE Trans. on System, Man, and Cybern.* Vol.28, No.3, 1998, pp. 371-391.
[3] P.Barany, P.Korondi, H.Hashimoto and M.Wada, Fuzzy Inversion and Rule Base Reduction, *IEEE Int. Conf. on Intell. Eng. Systems*, 1997, pp.301-306.
[4] H.R. Berenji and P.S. Khedkar, Learning and Tuning Fuzzy Logic Controllers Through Reinforcements, *IEEE Trans. on Neural Networks.* Vol. 3, 1992, pp.724-740.
[5] P.Craven, R.Sutton and M.Kwiesielewicz, Neurofuzzy Control of a Non-Linear Multivariable System, *IEE UKACC Int. Conf. on Control*, 1998, pp.531-536.
[6] F.Gomide and M. Figueiredo, Design of Fuzzy Systems Using Neurofuzzy Networks, *IEEE Trans. on Neural Networks.* Vol. 10, 1999, pp.815-827.
[7] F.Guély and P.Siarry, Gradient Descent Method for Optimizing Various Fuzzy Rule Bases, *Proc. 2nd IEEE Int. Conf. on Fuzzy Systems*, 1993, pp.1241-1246.
[8] F.Guély and P.Siarry, A Genetic Algorithm for Optimizing Takagi-Sugeno Fuzzy Rule Bases, *Fuzzy Sets and Systems*, Vol.99, No. 1, 1998, pp.37-47.
[9] S.-M.Guu and C.-T. Pang, On the Asymptotic Stability of Free Fuzzy Systems, *IEEE Trans. on Fuzzy Systems*, Vol.7, No.4, 1999, pp. 467-468.
[10] S.-I.Horikawa, T.Furuhashi and T.Uchikawa, On Fuzzy Modeling Using Fuzzy Neural Network with the Back-Propagation Algorithm, *IEEE Trans. on Neural Networks*, Vol.3, No.5, 1992, pp.801-806.
[11] Z.Huaguang and Z.Bien, A Multivariable Fuzzy Generalized Predictive Control Approach and its Performance Analysis, *Proc. of the ACC*, Vol.4, 1998, pp.2276-2280.
[12] A.Jana, P.Yang, D.Auslander and R.Dave, Real Time Neuro-Fuzzy Control of a Nonlinear Dynamic System. *Biennial Conference of the North American Fuzzy Information Processing Society.* 1996, pp.210-214.

[13] J.-S.R.Jang, ANFIS: Adaptive Network-Based Fuzzy Inference System. *IEEE Trans. on System, Man, and Cybern.* Vol 23, 1993, pp. 665-685.

[14] J.-S.R.Jang, C.-T.Sun and E.Mizutani, *Neuro-Fuzzy and Soft Computing*, Prentice Hall, 1997.

[15] J.Joh, Y.-H.Chen and R.Langari, On the Stability Issues of Linear Takagi-Sugeno Fuzzy Models, *IEEE Trans. on Fuzzy Systems*, Vol.6, No.3, 1998, pp-402-410.

[16] R.M.Kandadai and J.M.Tien, Knowledge-base Generating Hierarchical Fuzzy-Neural Controller. *IEEE Trans. on Neural Networks*. Vol. 8, 1997, pp.1531-1541.

[17] A.Kandel and Y.-Q. Zhang, Compensatory Neurofuzzy Systems with Fast Learning Algorithms. *IEEE Trans. on Neural Networks.* Vol. 9, 1998, pp.83-105.

[18] H.-J.Kang, H.Son, C.Kwon and M.Park, A New Approach to Adaptive Fuzzy Control, *Proc. of the IEEE Int. Conf. on Fuzzy Systems*, Vol.1, 1998, pp.264-267.

[19] B.Kosko, *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Inelligence*, Prentice Hall, 1992.

[20] J.Liao and M.J.Er, Stability Analysis and Systematic Design of Fuzzy Controllers with Simplified Linear Control Rules, *Proc. of the 38° IEEE Conf. on Decision and Control*, 1999, pp.4864-4865.

[21] E.H.Mamdani and S.Assilian, An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller, *Int. Journal of Man-Machine Studies*, Vol.7, No.1, 1975, pp.1-13.

[22] F.F.Mascioli and G.Martinelli, Constructive Approach to Neuro-Fuzzy Networks, *Signal Processing*, Vol.64, No.3, 1998, pp.347-358.

[23] T.M. McKinney and N.Ketharnavaz, Fuzzy Rule Generation Via Multi-Scale Clustering, *Proc. of the IEEE Int. Conf. on Systems, Man and Cybern.*, Vol.4, 1997, pp.3182-3187.

[24] D.Nauck, F.Klawonn and R.Kruse, *Foundations of Neuro-Fuzzy Systems*, John Wiley, 1997.

[25] A.Nurnberger and R.Kruse, Neuro-Fuzzy Techniques Under Matlab/Simulink Applied to Real Plant. *Proceedings of the IEEE World Congress on Comput. Intelligence*, Vol .I, 1998, pp.572-576.

[26] K.M.Passino and H.N.Nounou, Fuzzy Model Predictive Control: Techniques, Stability Issues, and Examples, *Proc. of the IEEE Int. Symposium on Intelligent Control*, 1999, pp.423-428.

[27] L.Schnitman and T.Yoneyama, A Method for Learning Membership Functions in Mamdani Fuzzy Models. *Proceedings of the XIII Brasilian Congress on Automation.* 2000.

[28] L.Schnitman and T.Yoneyama, An Efficient Implementation of a Learning Method for Mamdani Fuzzy Models, *Proceedings of the VIth Brasilian Simposium on Neural Networks.* Vol.1, 2000, pp.38-43

[29] L.Schnitman, J.A.M. Fellippe de Souza and T.Yoneyama, A New Mamdani-Like Fuzzy Structure. *WSES Fuzzy Sets and Fuzzy Systems.* February 2001.

[30] Z.Shan, H.-M.Kim and F.-Y.Wang, Plant Identification and Performance Optimization for Neuro-Fuzzy Networks, *Proc. of the IEEE Int. Conf. on Systems, Man and Cybern.*, Vol.4, 1996, pp.2607-2612.

[31] K.Sio and C.Lee, Stability of Fuzzy *PID* Controllers, *IEEE Trans. on Syst., Man, and Cybern. Part A: Systems and Humans*, Vol.28, No.4, 1998, pp.490-495.

[32] M.Sugeno and G.Kang, Structure Identification of Fuzzy Model, *Fuzzy Sets and Systems*, Vol.28, 1986, pp.329-346.

[33] V.Tahani and F.Sheikholeslam, Stability Analysis and Design of Fuzzy Control Systems, *IEEE Int. Conf. on Fuzzy Systems*, Vol.1, 1998, pp.456-461.

[34] T.Takagi and M.Sugeno, Fuzzy Identification on Systems and its Applications to Modeling and Control, *IEEE Trans. on Systems, Man, and Cybern.*, Vol.15, 1985, pp.116-132.

[35] H.Ying, An Analytical Study on Structure, Stability and Design of General Nonlinear Takagi-Sugeno Fuzzy Control Systems, *Automatica*, Vol.34, No.12, 1998, pp.1617-1623.

[36] H.Ying, General Takagi-Sugeno Fuzzy Systems with Simplified Linear Rule Consequent are Univrsal Controllers, Models and Filters. *Information Science*. Vol.108, 1998, pp.91-107.

[37] L.A.Zadeh, Fuzzy Sets, *Information and Control*, Vol.8, 1965, pp.338-353.