A new on-line self-constructing neural fuzzy network

Andrés Ferreyra, José de Jesús Rubio

Abstract—In this paper, we propose a new on-line self-constructing neural fuzzy network. Structure and parameter learning are updated at the same time in our algorithm, because there is no difference between them. It generates groups with a given radius. The center is updated in order to get a nearest one to the incoming data in each iteration, in this way, it does not generate many rules and it does not need to prune them. We give a time varying learning rate for backpropagation training. We use extended Kalman filter to train the center of sets in the THEN part. We proved the stability in both cases.

I. INTRODUCTION

Both neural networks and fuzzy logic are universal estimators which can approximate any nonlinear function to any prescribed accuracy, if sufficient hidden neurons or fuzzy rules are available. Recent results show that the fusion procedure of these two different technologies seems to be very effective for nonlinear systems identification [2]. In the last few years, the application of fuzzy neural networks to nonlinear systems identification has been a very active area [9],[10]. Fuzzy modeling involves structure and parameters identification. The second one is usually addressed by some gradient descent variant, e.g., the least square algorithm and backpropagation.

Structure identification is used to select fuzzy rules, it often lies on a substantial amount of heuristic observation to express proper strategy’s knowledge. It often tackled by off-line trial and error approaches, like the unbiasedness criterion [11]. Several approaches generate fuzzy rules from numerical data. One of the most common methods for structure initialization is uniform partitioning of each input variable into fuzzy sets, resulting to a fuzzy grid. This approach is followed in ANFIS [5]. In [1] the TKS was used for designing various neurofuzzy identifiers. This approach consists of two learning phases: structure learning which involves to find the main input variables of all the possible, specifying the membership functions, the partition of the input space and determining the number of fuzzy rules, and the second one, the parameter learning which involves the unknown parameters determination and the optimization of the ready existing ones in the model, using some optimization method based on the linguistic information from the human expert and from the numeric data obtained from the actual system to be modeled. These two learning phases are interrelated, and none of them can be independent from the other one. Traditionally, these phases are done sequentially, the parameter updating is employed after the structure is decided.

It is suitable only for off-line operations. Most of structure identification methods are based on data clustering, such as fuzzy C-means clustering [14],[16], mountain clustering [10], subtractive clustering [3]. These approaches require that all input-output data are ready before starting to identify the plant. So the structure identification approaches are off-line.

There are a few on-line methods in the literature. In [6] the input space is partitioned according to an aligned clustering-based algorithm. After the number of rules is decided, the parameters are tuned by recursive least square algorithm, it is called SONFIN. In [7] it is the recurrent case of the above case, it is called RSONFIN. In [15] the input space is automatically partitioned into fuzzy subsets by adaptive resonance theory mechanism. Fuzzy rules that tend to give a high output error are split in two, by a specific fuzzy rule splitting procedure. In [8] is proposed a radius to make clustering updates. In [19] is considered each group as one rule, and each rule is trained by its group data, they give a time varying learning rate for backpropagation algorithm in order to prove the parameter learning error is stable.

In this paper, we propose a new on-line self-constructing neural fuzzy network named "Kalman cluster fuzzy" where structure and parameter learning are updated at the same time. There is no difference between them. Kalman cluster fuzzy generates groups with a given radius. Updating the center in order to get a nearest one to the incoming data in each iteration, there are not many rules generated and it is not needed to prune the rule. We give a time varying learning rate for backpropagation training in the case of the centers and the widths. Using extended Kalman filter to train the center of sets in the THEN part, we prove the stability in both cases.

II. FUZZY NEURAL NETWORKS FOR NONLINEAR IDENTIFICATION

Consider the following unknown discrete-time nonlinear system

\[ y(k-1) = f [X(k-1)] \]

where \( X(k-1) = \{y(k-2), \ldots, y(k-n-1), u(k-2), \ldots, u(k-m-1)\} = [x_1(k-1) \ldots x_N(k-1)] \in \mathbb{R}^N \)

\( (N = n+m) \) is the input vector, \( |u(k-1)|^2 \leq \pi \), \( y(k-1) \) is the output of the plant, \( f \) is generally a nonlinear smooth function \( f \in C^\infty \). A generic fuzzy model is presented as a collection of fuzzy rules in the following form (Mandani fuzzy model [16])

\[ R_j : \text{IF } x_1 \text{ is } A_{1,j} \text{ and } x_2 \text{ is } A_{2,j} \text{ and } \ldots x_N \text{ is } A_{N,j} \text{ THEN } v \text{ is } B_j \]
we use $M$ ($j = 1, 2 \ldots M$) fuzzy IF-THEN rules and $N$ fuzzy sets for each rule to perform a mapping from an input linguistic vector $X(k-1) = [x_1(k-1) \ldots x_N(k-1)]_{\mathbb{R}^N}$ ($N = n+m$) to an output linguistic scalar $y(k-1)$. $A_{1,j} \ldots A_{N,j}$ and $B_j$ are standard fuzzy sets. Each input variable $x_i$ has $N$ fuzzy sets. In the case of [14],[16] we know, by using product inference, center-average defuzzifier and center fuzzifier, called Sugeno fuzzy inference system with weighted average (FIS), the output of the fuzzy logic system can be expressed as

$$\hat{y}(k-1) = a(k-1)/b(k-1)$$

(3)

where $x_i(k-1)$ are the inputs of the system (1), $(i = 1 \ldots N)$, $c_{ij}(k-1)$ and $\sigma_{ij}(k-1)$ are the centers and the widths of the membership function of the IF part, respectively, $(j = 1 \ldots M)$, $v_j(k-1)$ is the center of the membership function of the THEN part. If we define [17]

$$\phi_j(k-1) = z_j(k-1)/b(k-1)$$

(4)

then (3) can be written as follows

$$\hat{y}(k-1) = \sum_{j=1}^{M} \phi_j(k-1)v_j(k-1) = \Phi^T(k-1)V(k-1)$$

(5)

where $V(k-1) = [v_1(k-1) \ldots v_M(k-1)]^T \in \mathbb{R}^M$ and $\Phi(k-1) = [\phi_1(k-1) \ldots \phi_M(k-1)]^T \in \mathbb{R}^M$.

Remark 1: The structure is similar to the given in [17],[18], where the training in all the parameters was done with the gradient. In this paper because of $c_{ij}(k-1)$ and $\sigma_{ij}(k-1)$ are nonlinear, we use the gradient and because of $v_j(k-1)$ is linear, we use the extended Kalman filter which is proven in [12] and [20] and that it is better than the gradient.

III. STRUCTURE IDENTIFICATION

Let $x_i(k-1)$ are newly incoming pattern, then we get

$$p(k-1) = \max_{1 \leq j \leq M} z_j(k-1)$$

(6)

If $p(k-1) < r$, then a new rule is generated (each rule corresponds to each center) and $M = M + 1$ where $r$ is a selected radius, $r \in (0,1)$. Once a new rule is generated, the next step is to assign initial centers and widths of the corresponding membership functions; a new density ($d_{i,M+1}(k) = 1$ which indicates the number of elements for each rule) is generated for this rule.

$$c_{i,M+1}(k) = x_i(k-1) \quad \sigma_{i,M+1}(k) = \text{rand} \in (0,1) \quad d_{i,M+1}(k) = 1$$

(7)

If $p(k-1) \geq r$, then a rule is not generated and in the case that $z_j(k-1) = p(k-1)$ we have the winner rule $j^*$, the centers and density of this rule are updated as

$$c_{i,j^*}(k) = c_{i,j^*}(k-1) + \frac{1}{1+e^{c_{i,j^*}(k-1)-x_i(k-1)}} (x_i(k-1)-c_{i,j^*}(k-1))$$

$$d_{i,j^*}(k) = d_{i,j^*}(k-1) + 1$$

(8)

Remark 2: The structure identification is a little similar to the given in [6], [7], but they do not take the max of $z_j(k-1)$ as (6), this idea is taken from the competitive learning of ART recurrent neural network [4],[14] in order to get the winner rule (in the case of ART is the winner neuron). If the algorithm of [6], [7] does not generate a new rule, it does nothing. In this paper the center is updated as in (8) in order to get a nearest center to the incoming data in each iteration, in this way, it does not need to generate a new rule in each iteration, i.e., it does not generate many rules and it does not need to prune the rules (pruning rules is not convenient). This idea is similar to the updating of weights in the Kohonen recurrent neural network [4], (in this case they speak about weights and we speak about the center of the membership functions).

IV. PARAMETER IDENTIFICATION

We need the stability of parameters identification because this algorithm works on line. We first analyze the stability of centers and the widths of the membership functions of the IF part, later we analyze the stability of the centers of the membership functions of the THEN part.

We assume from [16] and [19] that fuzzy is a general approximator of nonlinear functions, then (1) can be written as

$$y(k-1) = a^*(k-1)/b^*(k-1) - \mu(k-1)$$

$$a^*(k-1) = \sum_{j=1}^{M} v_j^*(k-1) z_j^*(k-1)$$

$$b^*(k-1) = \sum_{j=1}^{M} z_j^*(k-1)$$

(9)

$$z_j^*(k-1) = \prod_{i=1}^{N} \exp \left\{ -\left( \frac{y_i(x_i(k-1) - c_{ij}(k-1))}{\sigma_{ij}(k-1)} \right)^2 \right\}$$

where $v_j^*(k-1)$, $c_{ij}(k-1)$ and $\sigma_{ij}(k-1)$ are unknown parameters which may minimize the modelling error $\mu(k-1)$. In the case of two independent variables, smooth function has Taylor formula as

$$f(x_1, x_2) = f(x_1, x_2) + \frac{\partial f(x_1, x_2)}{\partial x_1} (x_1 - x_1) + \frac{\partial f(x_1, x_2)}{\partial x_2} (x_2 - x_2) + \zeta(k-1)$$

(10)

where $\zeta(k-1)$ is the remainder of the Taylor formula. If we let $x_1$ and $x_2$ correspond $c_{ij}(k-1)$ and $\sigma_{ij}(k-1)$, $x_1$, $x_2$ correspond $c_{ij}^*(k-1)$ and $\sigma_{ij}^*(k-1)$, and define $\tilde{c}_{ij}(k-1) = c_{ij}(k-1) - c_{ij}^*(k-1)$, $\tilde{\sigma}_{ij}(k-1) = \sigma_{ij}(k-1) - \sigma_{ij}^*(k-1)$,
then apply Taylor formula to (3) and (9) and gives

$$\hat{y}(k-1) = y(k-1) + \frac{\partial \hat{y}(k-1)}{\partial \tilde{c}_{ij}(k-1)} \tilde{c}_{ij}(k-1) + \frac{\partial \hat{y}(k-1)}{\partial \tilde{\sigma}_{ij}(k-1)} \tilde{\sigma}_{ij}(k-1) + \zeta(k-1)$$

(11)

Using chain rule, we get

$$\frac{\partial \hat{y}(k-1)}{\partial \tilde{c}_{ij}(k-1)} = \frac{\partial \hat{y}(k-1)}{\partial y(k-1)} \frac{\partial y(k-1)}{\partial \tilde{c}_{ij}(k-1)} + \frac{\partial \hat{y}(k-1)}{\partial \sigma_{ij}(k-1)} \frac{\partial \sigma_{ij}(k-1)}{\partial \tilde{c}_{ij}(k-1)}$$

$$= \frac{\partial \hat{y}(k-1)}{\partial y(k-1)} \frac{\partial y(k-1)}{\partial z_{ij}(k-1)} \frac{\partial z_{ij}(k-1)}{\partial \tilde{c}_{ij}(k-1)} + \frac{\partial \hat{y}(k-1)}{\partial \sigma_{ij}(k-1)} \frac{\partial \sigma_{ij}(k-1)}{\partial \tilde{c}_{ij}(k-1)}$$

$$= 2 \nu_{ij} \frac{z_{ij}(k-1) - \sigma_{ij}(k-1)}{b(k-1) \sigma_{ij}(k-1)} \tilde{c}_{ij}(k-1) + \frac{\partial \hat{y}(k-1)}{\partial \sigma_{ij}(k-1)} \frac{\partial \sigma_{ij}(k-1)}{\partial \tilde{c}_{ij}(k-1)}$$

$$= \frac{\partial \hat{y}(k-1)}{\partial \tilde{c}_{ij}(k-1)} = \frac{\partial \hat{y}(k-1)}{\partial y(k-1)} \frac{\partial y(k-1)}{\partial z_{ij}(k-1)} \frac{\partial z_{ij}(k-1)}{\partial \tilde{c}_{ij}(k-1)} + \frac{\partial \hat{y}(k-1)}{\partial \sigma_{ij}(k-1)} \frac{\partial \sigma_{ij}(k-1)}{\partial \tilde{c}_{ij}(k-1)}$$

(13)

we define the identification error as

$$e(k-1) = \hat{y}(k-1) - y(k-1)$$

(12)

so

$$\hat{y}(k-1) = y(k-1) + \frac{2 \nu_{ij} \frac{z_{ij}(k-1) - \sigma_{ij}(k-1)}{b(k-1) \sigma_{ij}(k-1)} \tilde{c}_{ij}(k-1) + \frac{\partial \hat{y}(k-1)}{\partial \sigma_{ij}(k-1)} \frac{\partial \sigma_{ij}(k-1)}{\partial \tilde{c}_{ij}(k-1)}}{b(k-1) \sigma_{ij}(k-1)}$$

(13)

If we define:

$$D_1(k-1) = \frac{2 \nu_{ij} \frac{z_{ij}(k-1) - \sigma_{ij}(k-1)}{b(k-1) \sigma_{ij}(k-1)} \tilde{c}_{ij}(k-1) + \frac{\partial \hat{y}(k-1)}{\partial \sigma_{ij}(k-1)} \frac{\partial \sigma_{ij}(k-1)}{\partial \tilde{c}_{ij}(k-1)}}{b(k-1) \sigma_{ij}(k-1)}$$

$$D_2(k-1) = \frac{2 \nu_{ij} \frac{z_{ij}(k-1) - \sigma_{ij}(k-1)}{b(k-1) \sigma_{ij}(k-1)} \tilde{c}_{ij}(k-1) + \frac{\partial \hat{y}(k-1)}{\partial \sigma_{ij}(k-1)} \frac{\partial \sigma_{ij}(k-1)}{\partial \tilde{c}_{ij}(k-1)}}{b(k-1) \sigma_{ij}(k-1)}$$

(14)

then (13) is

$$\hat{y}(k-1) = y(k-1) + D_1(k-1)c_{ij}(k-1) + D_2(k-1)e_{ij}(k-1) + \zeta(k-1)$$

(15)

In order to assure the stability of identification we use the following learning law to update the weights of neural identifier

$$c_{ij}(k) = c_{ij}(k-1) - \eta(k-1)D_1(k-1)c_{ij}(k-1) \frac{e(k-1)}{1 + q(k-1)}$$

$$\sigma_{ij}(k) = \sigma_{ij}(k-1) - \eta(k-1)D_2(k-1)e_{ij}(k-1) \frac{e(k-1)}{1 + q(k-1)}$$

(16)

where $j = 1,...,M$, $i = 1,...,N$ and $D_1(k-1)$ and $D_2(k-1)$ are given in (14). Note that $D_1(k-1)$ are auxiliary terms to minimize the notation, but the right way is to substitute $D_1(k-1)$ from (14) into (16). The dead-zone is applied to $\eta(k-1)$ as

$$\eta(k-1) = \begin{cases} \frac{\eta}{1 + q(k-1)} & \text{if } e^2(k-1) \geq \frac{\gamma^2}{1 - \eta} \\ 0 & \text{if } e^2(k-1) < \frac{\gamma^2}{1 - \eta} \end{cases}$$

(17)

where $\gamma$ is the upper bound of $\zeta(k-1)$, $q(k-1) = D_2^2(k-1)$, and $0 < \eta \leq 1$.

The following theorem gives the stability of neural identification in the case of centers and widths of the IF part. This theorem is similar to the given in [13].

**Theorem 1:** If we use a fuzzy neural model (3) to identify the nonlinear system (1), the learning law (14),(16) with dead-zone (17) makes the identification stable, i.e., the identification error $e(k-1)$ is bounded and satisfies:

$$\lim_{k \to \infty} e^2(k-1) = \frac{\gamma^2}{1 - \eta}$$

(18)

**Proof:** We select the following Lyapunov function

$$L_1(k-1) = \frac{\gamma^2}{1 - \eta}$$

(19)

By updating (16), we have

$$e_{ij}(k) = \tilde{c}_{ij}(k-1) - \eta(k-1)D_1(k-1)e(k-1) + \tilde{\sigma}_{ij}(k-1) - \eta(k-1)D_2(k-1)e(k-1)$$

Now we calculate $\Delta L_1(k)$

$$\Delta L_1(k) = [\tilde{c}_{ij}(k-1) - \eta(k-1)D_1(k-1)e(k-1)]^2 - \tilde{\sigma}_{ij}(k-1) - \eta(k-1)D_2(k-1)e(k-1)]^2$$

$$= \eta^2(k-1) \left\{ D_1^2(k-1) + D_2^2(k-1) \right\} e^2(k-1) - 2\eta(k-1)D_1(k-1)e_{ij}(k-1) + D_2(k-1)e_{ij}(k-1) + \tilde{\sigma}_{ij}(k-1)$$

(20)

Substituting (15) into the last term of (20) and using (17) gives

$$\Delta L_1(k) \leq \eta^2(k-1) \left\{ 1 + D_1^2(k-1) + D_2^2(k-1) \right\} e^2(k-1)$$

$$- \eta(k-1)e^2(k-1) - \eta(k-1)e^2(k-1) + \tilde{\sigma}_{ij}(k-1)$$

$$\Delta L_1(k) \leq \eta^2(k-1) \left\{ 1 + q(k-1) \right\} e^2(k-1)$$

(21)
From the dead-zone, \( e^2(k-1) \geq \frac{n^2}{1-n} \) and \( \eta(k-1) > 0 \), \( \Delta L_1(k-1) \leq 0 \). \( L_1(k) \) is bounded. If \( e^2(k-1) < \frac{n^2}{1-n} \), from (17) we know \( \eta(k-1) = 0 \), all of the weights are not changed, they are bounded, so \( L_1(k) \) is bounded.

When \( e^2(k-1) \geq \frac{n^2}{1-n} \), summarize (21) from 2 to \( T \)

\[
\sum_{k=2}^{T} \eta(k-1) \left[ (1-\eta) e^2(k-1) - \frac{n^2}{1-n} \right] \leq L_1(1) - L_1(T)
\]

(22)

Since \( L_1(T) \) is bounded and using \( \eta(k-1) = \frac{n}{1+q(k-1)} > 0 \)

\[
\lim_{k \to \infty} \sum_{k=2}^{T} \left( \frac{n}{1+q(k-1)} \right) \left[ (1-\eta) e^2(k-1) - \frac{n^2}{1-n} \right] < \infty
\]

(23)

Because \( e^2(k-1) \geq \frac{n^2}{1-n} \),

\[
\left( \frac{n}{1+q(k-1)} \right) \left[ (1-\eta) e^2(k-1) - \frac{n^2}{1-n} \right] \geq 0,
\]

so

\[
\lim_{k \to \infty} \left( \frac{n}{1+q(k-1)} \right) \left[ (1-\eta) e^2(k-1) - \frac{n^2}{1-n} \right] = 0
\]

(24)

Because \( L_1(k-1) \) is bounded, so \( q(k-1) < \infty \), and as \( \frac{n}{1+q(k-1)} > 0 \)

\[
\lim_{k \to \infty} (1-\eta) e^2(k-1) = \frac{n^2}{1-n}
\]

(25)

That is (18). When \( e^2(k-1) < \frac{n^2}{1-n} \), it is already in this zone.

Remark 3: The normal learning (14),(16) has similar form as backpropagation [16], the only difference is that we use normalizing learning rate \( \eta(k-1) \), [16] use fixed learning rate. The time-varying learning rate can assure the identification stable. This learning rate is easy to get, no any prior information is required, for example we may select \( \eta = 1 \).

Now, we prove the stability of the centers of the THEN part.

From (4) and (5), (9) can be written as

\[
g(k-1) = \Phi^T(k-1) V(k-1) + \mu(k-1)
\]

(26)

where \( V^*(k-1) \) is the optimal weight which can minimize the modeling error \( \mu(k-1) \).

From (5), (12) and (26) we have

\[
e(k-1) = \Phi^T(k-1) \tilde{V}(k-1) + \mu(k-1)
\]

(27)

where \( \tilde{V}(k-1) = V^*(k-1) - V(k-1) \). We modify the extended Kalman filter [12],[20] as dead-zone Kalman filter

\[
V(k) = V(k-1) - \frac{1}{\pi_{k-1}} P_{k-1} \Phi(k-1) s_{k-1}
\]

\[
P_k = R_1 + P_{k-1} - \frac{1}{\pi_{k-1}} P_{k-1} \Phi(k-1) \Phi^T(k-1) P_{k-1}
\]

\[
g(k) = \Phi^T(k-1) V(k-1)
\]

(28)

where

\[
s_{k-1} = \begin{cases} 
  e(k-1) & \text{if } e^2(k-1) \geq \frac{n^2}{1-n} \text{ AND } E \\
  0 & \text{if } e^2(k-1) < \frac{n^2}{1-n} \text{ OR } F 
\end{cases}
\]

\[
E = \left\| \frac{1}{\pi_{k-1}} P_{k-1} \Phi(k-1) \Phi^T(k-1) P_{k-1} \right\| \geq \| R_1 \|
\]

\[
F = \left\| \frac{1}{\pi_{k-1}} P_{k-1} \Phi(k-1) \Phi^T(k-1) P_{k-1} \right\| < \| R_1 \|
\]

(30)

\[
R_{k-1} = R_2 + \Phi^T(k-1) P_{k-1} \Phi(k-1) - \pi\]

(31)

First, we consider the case \( e^2(k-1) \geq \frac{n^2}{1-n} \) and

\[
\frac{1}{\pi_{k-1}} P_{k-1} \Phi(k-1) \Phi^T(k-1) P_{k-1} \geq R_1
\]

So \( s_{k-1} = e(k-1) \). Substituting (27) into (28) gives

\[
g(k) = \tilde{V}(k-1) - \frac{1}{\pi_{k-1}} P_{k-1} \Phi(k-1) \Phi^T(k-1) \tilde{V}(k-1)
\]

(32)

From (28), (27), and (32) we have

\[
g(k) = \tilde{V}(k-1) - \frac{1}{\pi_{k-1}} P_{k-1} \Phi(k-1) \left( \mu(k-1) \right)
\]

(33)

From (28) we know \( P_k = \pi_{k-1} + R_1 \).

So \( \pi_k = \pi_{k-1} - \frac{1}{\pi_{k-1}} P_{k-1} \Phi(k-1) \Phi^T(k-1) P_{k-1} \).

\[
\pi_{k-1} P_{k-1}^{-1} \tilde{V}(k) = P_{k-1}^{-1} \tilde{V}(k-1) - \frac{1}{\pi_{k-1}} \pi_{k-1}^{-1} P_{k-1} \Phi(k-1) \mu(k-1)
\]

(34)

From (28) into (31) gives

\[
\Delta L_2(k) \leq \left( \tilde{V}(k) - \tilde{V}(k-1) \right)^T P_{k-1}^{-1} \tilde{V}(k-1)
\]

(35)
here \( \left[ \widetilde{V}(k) - \widetilde{V}(k-1) \right]^T \) is substituted by (32), (35) is
\[
\Delta L_2(k) \leq -\frac{1}{R_{k-1}} \mu(k-1) \Phi(k-1) \Phi^T(k-1) \tilde{V}(k-1) - \frac{\mu(k-1)}{R_{k-1}} \Phi^T(k-1) \tilde{V}(k-1) - \frac{1}{R_{k-1}} \tilde{V}(k) \mathbf{P}^{-1}_{k-1} \Phi(k-1) \mu(k-1) \]
(36)

The last term of (36) is
\[
\frac{\mu(k-1)}{R_{k-1}} \Phi^T(k-1) \tilde{V}(k-1) \mathbf{P}^{-1}_{k-1} \Phi(k-1) \]
\[
= \frac{\mu(k-1)}{R_{k-1}} \Phi^T(k-1) P_{k-1} \mathbf{P}^{-1}_{k-1} \tilde{V}(k) \]
\[
= \frac{\mu(k-1)}{R_{k-1}} \Phi^T(k-1) P_{k-1} \mathbf{P}^{-1}_{k-1} \tilde{V}(k) \]
\[
= \frac{\mu(k-1)}{R_{k-1}} \Phi^T(k-1) P_{k-1} \mathbf{P}^{-1}_{k-1} \Phi(k-1) \]

Now we apply the matrix inversion lemma to
\[
\mathbf{P}^{-1}_{k} = P_{k-1} - \frac{1}{R_{k-1}} P_{k-1} \Phi(k-1) \Phi^T(k-1) P_{k-1}
\]
\[
(\mathbf{A} + CD)^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1} B \left( \mathbf{DA}^{-1} B + C^{-1} \right)^{-1} \mathbf{DA}^{-1}
\]
(37)

with
\[
\mathbf{A}^{-1} = P_{k-1}, \quad B = \Phi(k-1), \quad C^{-1} = R_2,
\]
\[
D = \Phi^T(k-1), \quad R_{k-1} = R_2 + \Phi^T(k-1) P_{k-1} \Phi(k-1),
\]
so \( \mathbf{P}^{-1}_{k-1} = P_{k-1} - \frac{1}{R_{k-1}} \Phi(k-1) \Phi^T(k-1) P_{k-1} \Phi(k-1) \).

(38)

\[
\Delta L_2(k) \leq -\frac{1}{R_{k-1}} \tilde{V}(k-1) \Phi(k-1) \Phi^T(k-1) \tilde{V}(k-1) - \frac{\mu(k-1)}{R_{k-1}} \Phi^T(k-1) \tilde{V}(k-1) - \frac{1}{R_{k-1}} \tilde{V}(k) \mathbf{P}^{-1}_{k-1} \Phi(k-1) \mu(k-1) \]
\[
= -\frac{1}{R_{k-1}} \left[ \tilde{V}(k-1) \Phi(k-1) \Phi^T(k-1) \tilde{V}(k-1) + 2 \mu(k-1) \Phi^T(k-1) \tilde{V}(k-1) + \mu^2(k-1) \right] \]
\[
+ \frac{1}{R_{k-1}} \left( P_{k-1} + \frac{1}{R_2} \Phi(k-1) \Phi^T(k-1) P_{k-1} \Phi(k-1) \right) \mu^2(k-1)
\]
\[
= -\frac{1}{R_{k-1}} \left[ \Phi^T(k-1) \tilde{V}(k-1) + \mu(k-1) \right]^2 \]
\[
+ \frac{1}{R_{k-1}} \left[ 1 + \frac{\Phi^T(k-1) P_{k-1} \Phi(k-1)}{R_2 + \Phi^T(k-1) P_{k-1} \Phi(k-1)} \right] \mu^2(k-1)
\]
\[
+ \frac{1}{R_{k-1}} \left( \Phi^T(k-1) P_{k-1} \Phi(k-1) \right) \mu^2(k-1)
\]
(39)

Using (27), (38) tells us
\[
\Delta L_2(k) \leq -\frac{1}{R_{k-1}} e^2(k) + \frac{1}{R_{k-1}} e^2(k) + \left( \frac{1}{R_{k-1}} (1 + 1 + 1) \right) \mu^2(k-1)
\]
\[
\Delta V(k) \leq -\frac{1}{R_{k-1}} e^2(k) + \frac{1}{R_{k-1}} e^2(k) + \left( \frac{1}{R_{k-1}} (1 + 1 + 1) \right) \mu^2(k-1)
\]

So
\[
\Delta L_2(k) \leq -\frac{1}{R_{k-1}} \left[ e^2(k-1) - 3\mu^2 \right]
\]
(40)

where \( |\mu(k-1)| < \bar{\mu} \).

Because \( e^2(k-1) \geq 3\mu^2 \) and \( R_{k-1} > 0 \), so \( \Delta V(k) \leq 0 \).

Thus
\[
\tilde{V}(k-1) P_{k-1} \mathbf{P}^{-1}_{k-1} \tilde{V}(k-1) \leq \tilde{V}(1) P_{1}^{-1} \tilde{V}(1) \]
\[
\leq \lambda_{\text{max}}(P_{1}^{-1}) \left\| \tilde{V}(1) \right\|^2
\]

From (28) we know
\[
P_k = P_{k-1} - \frac{1}{\bar{R}_{k-1}} P_{k-1} \Phi(k-1) \Phi^T(k-1) P_{k-1} + R_1,
\]

because \( \left\| \frac{1}{\bar{R}_{k-1}} P_{k-1} \Phi(k-1) \Phi^T(k-1) P_{k-1} \right\| \geq ||R_1||, xP_k x \geq xP_{k-1} x \)

\[
\lambda_{\text{min}}(P_{1}^{-1}) \left\| \tilde{V}(1) \right\|^2 \leq \tilde{V}(k-1) P_{k-1} \mathbf{P}^{-1}_{k-1} \tilde{V}(k-1) \leq \tilde{V}(1) P_{1}^{-1} \tilde{V}(1)
\]

we choose \( \lambda_{\text{min}}(P_{1}^{-1}) \neq 0 \), so

\[
\left\| \tilde{V}(k-1) \right\|^2 \leq \frac{\lambda_{\text{max}}(P_{1}^{-1})}{\lambda_{\text{min}}(P_{1}^{-1})} \left\| \tilde{V}(1) \right\|^2
\]

\( V(k-1) \) is bounded. On the other side, if \( e^2(k-1) < 3\mu^2 \) or \( \left\| \frac{1}{\bar{R}_{k-1}} P_{k-1} \Phi(k-1) \Phi^T(k-1) P_{k-1} \right\| < ||R_1||, s_{k-1} = 0 \), \( V(k) = V(k-1), V(k) \) is bounded. For all the cases, the weights \( V(k) \) are bounded.

When \( e^2(k-1) \geq 3\mu^2 \), summarize (39) from 2 to T
\[
\sum_{k=2}^{T} \frac{1}{R_{k-1}} \left[ e^2(k-1) - 3\mu^2 \right] \leq L_2(1) - L_2(T)
\]

Since \( L_2(T) \) is bounded and using \( \frac{1}{R_{k-1}} > 0 \)
\[
\lim_{T \to \infty} \sum_{k=2}^{T} \left( \frac{1}{R_{k-1}} \right) \left[ e^2(k-1) - 3\mu^2 \right] < \infty
\]

Because \( e^2(k-1) \geq 3\mu^2 \), \( \left( \frac{1}{R_{k-1}} \right) \left[ e^2(k-1) - 3\mu^2 \right] \geq 0 \),

so
\[
\lim_{k \to \infty} \left( \frac{1}{R_{k-1}} \right) \left[ e^2(k-1) - 3\mu^2 \right] = 0
\]

Because \( L_2(k-1) \) is bounded, so \( R_{k-1} < \infty \), and as \( \frac{1}{\bar{R}_{k-1}} > 0 \)
\[
\lim_{k \to \infty} e^2(k-1) = 3\mu^2
\]

(42)

That is (29). When \( e^2(k-1) < 3\mu^2 \), it is already in this zone.
V. ALGORITHM PROPOSED

The final algorithm proposed is the next:

1) For the first data \(k = 1\), \(M = 1\), \(v_1(1) = y(1)\), \(d_{s1}(1) = 1\), \((i = 1 \ldots N)\), \(\sigma_{s1}(1) = x_i(1)\), \(\sigma_{s1}(1) = \text{rand} \in (0,1)\), \(P_1 = 100 \in \mathbb{R}^{1x1}\), \(R_1 = \alpha \in \mathbb{R}^{1x1}\) and \(R_2 = \beta\).

2) For the other data \(k \geq 2\), evaluate \(z_j(k-1)\) and \(b(k-1)\) with (3), evaluate \(\tilde{y}_j(k-1)\) with (4) and (5), evaluate \(p(k-1)\) with (6), update \(\sigma_{ij}(k)\) and \(\sigma_{s1}(k)\) with (14) (16) and (17). Note that \(D_i(k-1)\) are auxiliary terms to minimize the notation, but the right is to substitute \(D_i(k-1)\) from (14) into (16).

3) If \(p(k-1) < r\), then a new rule is generated \((M = M + 1)\) where \(r \in (0,1)\), assign initial values to \(c_{i,M+1}(k)\), \(\sigma_{i,M+1}(k)\), \(y_{M+1}(k)\) and \(d_{i,M+1}(k)\) with (7), \(P_{M+1,M+1,k} = 100 \in P_k \in \mathbb{R}^{M+1xM+1}\), and \(\alpha = 100 \in R_1 \in \mathbb{R}^{M+1xM+1}\), go to 2.

4) If \(p(k-1) \geq r\), then a rule is not generated and in the case that \(z_j(k-1) = p(k-1)\) we have the winner rule \(j^*\), the value of \(c_{ij^*}(k)\) and \(d_{ij^*}(k)\) of this rule are updated with (8), go to 2.

VI. SIMULATION

In this section, the suggested on-line self organized algorithm is applied to nonlinear system identification.

**Example 1** Consider the nonlinear system given in [16] and [19]:

\[ y(k) = 0.52 + 0.1x_1 + 0.28x_2 - 0.06x_1x_2 \]  \( (44) \)

with \(x_1(k) = \sin^2(10/k)\) and \(x_2(k) = \cos^2(10/k)\). The parameters of our algorithm are \(\alpha = \beta = 0.01\), \(r = 0.1\). We compare our algorithm called Kalman cluster fuzzy with the version given in [19] called cluster fuzzy (in this case the gradient is used to train parameters and they do not use (8)) and with the version given in [16] called fuzzy (in this case they do not cluster but use gradient with constant learning rate). The identification is given in Fig.1 and the least mean square for fuzzy is 0.0026, for cluster fuzzy is 1.5832x10^{-4} and for Kalman cluster fuzzy is 1.1921x10^{-5}. In Fig.2 is given the membership functions of the Kalman cluster fuzzy algorithm. The cluster fuzzy algorithm generates 5 rules and the Kalman cluster fuzzy algorithm generates 2 rules, it is because Kalman cluster fuzzy uses (8).

**Example 2** Consider the nonlinear system:

\[ y(k) = \frac{F_1}{F_2} \]  \( (45) \)

\[ F_1 = y^2(k-3) + y^2(k-2) + y^2(k-1) + \tanh(u(k)) + 1 \]

\[ F_2 = y^2(k-3) + y^2(k-2) + y^2(k-1) + 1 \]

where \(u(k) = 0.6\sin(3\pi k Ts) + 0.2\sin(4\pi k Ts) + 1.2\sin(5\pi k Ts)\), \(Ts = 0.01\), \(x_1(k) = y(k-1)\), \(x_2(k) = y(k-2)\), \(x_3(k) = y(k-3)\), \(x_4(k) = u(k)\). The parameters of our algorithm are \(\alpha = \beta = 0.01\), \(r = 1x10^{-6}\).

Again, we compare the Kalman cluster fuzzy algorithm. The identification is given in Fig.3 and the least mean square for fuzzy is 0.0018, for cluster fuzzy is 0.0019 and for Kalman cluster fuzzy is 2.2x10^{-5}. In Fig.4 is given the membership functions of the Kalman cluster fuzzy algorithm. The cluster fuzzy algorithm generates 6 rules and the Kalman cluster fuzzy algorithm generates 2 rules, it is because Kalman cluster fuzzy uses (8).

VII. CONCLUSION

In the proposed on-line self-constructing neural fuzzy network, structure and parameter learning are updated at the same time without making difference between them. It generates groups with a given radius. The center is updated in order to get a nearest one to the incoming data in each iteration, in this way, it does not generate many rules and it does not need to prune them. We give a time varying learning rate for backpropagation training in the case of the centers and the widths in the membership functions of the IF part.
This assures an stability in identification; the learning rate is easy to obtain due to it is not necessary prior information, so we can select $\eta = 1$. And using extended Kalman filter to train the center in the membership functions of the THEN part we assure the bounded of identification error and weights of the neural networks.

Simulation results show that vantages of the proposed algorithm are the reduction of the number rules obtained, and that the value of the least mean square is the smallest one of the obtained from the other methods used for the comparison.

REFERENCES


