Proceedings of the
44th IEEE Conference on Decision and Control, and
the European Control Conference 2005
Seville, Spain, December 12-15, 2005

System identification using hierarchical fuzzy neural networks with stable learning algorithms

Wen Yu and Marco A. Moreno-Armendariz

Abstract— **Hierarchical fuzzy neural networks can use less rules to model nonlinear system with high accuracy. But the structure is very complex, the normal training for hierarchical fuzzy neural networks is difficult to realize. In this paper we use backpropagation-like approach to train the membership functions. The new learning schemes employ a time-varying learning rate that is determined from input-output data and model structure. Stable learning algorithms for the premise and the consequence parts of fuzzy rules are proposed. The calculation of the learning rate does not need any prior information such as estimation of the modeling error bounds. The new algorithms are very simple, we can even train each sub-block of the hierarchical fuzzy neural networks independently.**

I. INTRODUCTION

Both neural networks and fuzzy logic are universal estimators, they can approximate any nonlinear function to any prescribed accuracy, provided that sufficient hidden neurons and fuzzy rules are available. Resent results show that the fusion procedure of these two different technologies seems to be very effective for nonlinear systems identification [1][2][8]. Gradient descent and backpropagation are always used to adjust the parameters of membership functions (fuzzy sets) and the weights of defuzzification (neural networks) for fuzzy neural networks. Slow convergence and local minimum are main drawbacks of these algorithms [9]. Some modifications were derived in recently published literatures. [3] suggested a robust backpropagation law to resist the noise effect and reject errors drift during the approximation. [1] used B-spline membership functions to minimize a robust object function, their algorithm can improve convergence speed. In [18] structure and parameters of fuzzy neural systems were determined by RBF neural networks.

In the design of the fuzzy systems is common to use a table look-up approach, which is a time-consuming task. Especially when the number of inputs and membership functions are huge, the number of fuzzy rules increase exponentially. The huge rule base would be overload the memory and make the fuzzy system very hard to implement. Generally n input variables and m membership functions for each variable, neuro-fuzzy systems require m^n rules. This phenomenon is called "curse of dimensionality". In order to deal with the rule-explosion problem, a number of low-dimensional fuzzy systems in a hierarchical form are consisted, instead of a single high-dimensional fuzzy system. This is main idea of hierarchical fuzzy systems (HFS) [12][15]. It has been proven that hierarchical fuzzy systems also universal estimators [17]. In [19] a hierarchical prioritized structure are able to introduce exceptions to more general rules by giving them a priority and introducing them to a higher level. The lowest level contains de default rules about the relationship being modeled. The middle level contains rules based on aggregation of exceptions to these default rules. The highest level of the structure contains specific exceptions not accounted for by the rest of the model. In [7] a method using intermediate mapping variables as temporal variables is presented to avoid the designing of intermediate outputs. In [13] a type of HFS, called Hierarchical Classifying-Type Fuzzy System (HCTFS) is used instead of repetitive defuzzification process between subsystem layers, analyzes the computational complexity in terms of the mathematical process and electronic components used.

When we cannot decide the membership functions in prior, we should use the input/output data to train the parameters of the membership function, for example ANFIS [6] and gradient learning [15]. Even for a single fuzzy neural networks, the training algorithm is complex [22]. It is very difficult to realize learning for hierarchical fuzzy neural networks if we use normal learning [16]. By using backpropagation technique, gradient descent algorithms can be simplified for multilayer neural networks training. Nevertheless, can hierarchical fuzzy neural networks be trained by the similar technique? To the best of our knowledge, the training for hierarchical fuzzy neural system was still used the normal gradient algorithm [6][16].

The stability problem of fuzzy neural identification is very important in applications. It is well known that normal identification algorithms (for example, gradient descent and least square) are stable in ideal conditions. In the presence of unmodeled dynamics, they might become unstable. The lack of robustness of the parameter identification was demonstrated in [14] and became a hot issue in 1980s, when some robust modification techniques were suggested [5]. The learning procedure of fuzzy neural networks can be regarded as a type of parameter identification. Gradient descent and backpropagation algorithms are stable, if fuzzy neural models can match nonlinear plants exactly. However, some robust modifications must be applied to assure stability with respect to uncertainties. Projection operator is an effective tool to guarantee fuzzy modeling bounded [15]. It was also used by many fuzzy-neural systems [10]. Another general approach

Wen Yu is with the Departamento de Control Automatico, CINVESTAV-IPN, Av. IPN 2508, México D.F., 07360, México yuw@ctrl.cinvestav.mx

Marco A. Moreno-Armendariz is with Escuela de Ingeniería, Dirección de Posgrado e Investigación, LIDETEA, Universidad La Salle, Benjamin Franklin 47, Col. Condesa, México D.F, 06140, México. mmoreno@ci.ulsa.mx

0-7803-9568-9/05/$20.00 ©2005 IEEE

is to use robust adaptive techniques [5] in fuzzy neural modeling. For example, [16] applied a switch $\sigma-$modification to prevent parameters drift. By using passivity theory, we successfully proved that for continuous-time recurrent neural networks, gradient descent algorithms without robust modification were stable and robust to any bounded uncertainties [20], and for continuous-time identification they were also robustly stable [21]. Nevertheless, do hierarchical fuzzy neural networks have the similar characteristics?

In this paper backpropagation-like approach is applied to system identification via hierarchical fuzzy neural networks. Both the premise and the consequent membership functions are assumed to be unknown. The new algorithms are very simple, we can even train the parameters of each subblock independently. The new stable algorithms with time-varying learning rates are applied to hierarchical fuzzy neural networks. One example is given to illustrate the effectiveness of the suggested algorithms.

II. HIERARCHICAL FUZZY NEURAL NETWORKS

Consider following discrete-time nonlinear system

$$\begin{aligned} &y(k) = h\left[x\left(k\right)\right] = \Psi\left[X\left(k\right)\right] \\ &= \Psi\left[y\left(k-1\right), y\left(k-2\right), \cdots u\left(k-1\right), u\left(k-2\right), \cdots\right] \end{aligned} \tag{1}$$

where

$$X\left(k\right) = \left[y\left(k-1\right), y\left(k-2\right), \cdots u\left(k\right), u\left(k-1\right), \cdots\right]^T \tag{2}$$

A conventional fuzzy model with one output is presented as a collection of fuzzy rules in the following form (for example, Mamdani fuzzy model [15])

R^i : IF x_1 is A_1^i and $\cdots$ and x_n is A_n^i THEN $\widehat{y}_1$ is B^i

We use l $(i = 1, 2, \cdots, l)$ fuzzy IF-THEN rules to perform a mapping from an input linguistic vector $X = [x_1, \cdots, x_n] \in \Re^n$ to an output linguistic $\widehat{y}$. $A_1^i, \cdots, A_n^i$ and B^i are standard fuzzy sets. Each input variable $x_j\,(j = 1, 2, \ldots, n)$ has l_j fuzzy sets. In the case of full connection, $l = l_1 \times l_2 \times \cdots l_n$.

In order to design a conventional fuzzy system with a required accuracy, the number of rules has to increase exponentially with the number of input variables to the fuzzy system. Consider n input variables and m fuzzy sets for each input variable, then the number of rules in the fuzzy system is m^n. When n is large, the number of rules is a huge number. A serious problem facing fuzzy system applications is how to deal with this rule explosion problem. One approach to deal with this difficulty is use hierarchical fuzzy systems. This kind of systems have the nice property that the number of rules needed to construct the fuzzy system increases only *linearly* with the number of variables [16].To represent the output of each hierarchical block, the p_th level output $(p > 1)$ is

$$\widehat{y}_p = \frac{\sum_{i=1}^{l_p} w_p^i \left[\prod_{j=1}^{n_{p1}} \mu_{A_{p,j}^i}\left(x_{p,j}\right) \prod_{j=1}^{n_{p2}} \mu_{D_{p,j}^i}\left(\widehat{y}_{p-1,j}\right)\right]}{\sum_{i=1}^{l_p} \left[\prod_{j=1}^{n_{p1}} \mu_{A_{p,j}^i}\left(x_{p,j}\right) \prod_{j=1}^{n_{p2}} \mu_{D_{p,j}^i}\left(\widehat{y}_{p-1,j}\right)\right]} \tag{3}$$

where $\mu_{A_{p,j}^i}$ and $\mu_{D_{p,j}^i}$ are the membership functions of the fuzzy sets $A_{p,j}^i$ and $D_{p,j}^i$, w_p^i is the point at which $\mu_{B_p^i} = 1$

We use the following example to explain how to use the backpropagation technique for hierarchical fuzzy neural networks. Three fuzzy neural networks (FS1, FS2, FS3) form a hierarchical fuzzy neural networks.

For each subsystem , there are l fuzzy rule and n input, 1 output. If we use singleton fuzzifier, Mamdani implications, center average defuzzifier, the output can be expressed as (3), where $\mu_{A_j^i} = \exp\left[-\left(\frac{x_j - c_j^i}{\sigma_j^i}\right)^2\right]$ is the membership functions of the fuzzy sets A_j^i, $\overline{y}^i$ is the point at which $\mu_{B^i} = 1$, $\widehat{y}$ is the output of each fuzzy system. Let us define

$$\begin{aligned} z^i &= \prod_{j=1}^{n} \exp\left[-\left(\frac{x_j - c_j^i}{\sigma_j^i}\right)^2\right] \\ a &= \sum_{j=1}^{l} w^i z^i, \qquad b = \sum_{i=1}^{l} z^i \end{aligned} \tag{4}$$

So $\widehat{y} = \frac{a}{b}$. The the object of identification problem is to determine parameters w^i, c_j^i and σ_j^i such that the output of the fuzzy neural networks $\widehat{y}$ converge to the output of the plant y. Using the chain rule

$$\frac{\partial J}{\partial w^i} = \frac{\partial J}{\partial \widehat{y}} \frac{\partial \widehat{y}}{\partial w^i} = \left(\widehat{y} - y\right) \frac{\partial\left(\frac{a}{b}\right)}{\partial w^i} = \frac{\left(\widehat{y} - y\right)}{b} z^i \tag{5}$$

w^i is updated by

$$w^i\left(k+1\right) = w^i\left(k\right) - \eta \frac{z^i}{b}\left(\widehat{y} - y\right) \tag{6}$$

since $\frac{\partial e^x}{\partial x} = e^x$

$$\begin{aligned} &\frac{\partial J}{\partial c_j^i} = \frac{\partial J}{\partial \widehat{y}} \frac{\partial \widehat{y}}{\partial zi} \frac{\partial z^i}{\partial c_j^i} \\ &= \left(\widehat{y} - y\right)\left[\frac{w^i}{b} - \frac{a}{b^2}\right] z^i \left[2\frac{\left(x_j - c_j^i\right)}{\left(\sigma_j^i\right)^2}\right] \end{aligned} \tag{7}$$

So c_i^j is trained by

$$c_j^i\left(k+1\right) = c_j^i\left(k\right) - 2\eta\left(\widehat{y} - y\right) z^i \frac{\left(w^i - \widehat{y}\right)\left(x_j - c_j^i\right)}{b\left(\sigma_j^i\right)^2} \tag{8}$$

Similar

$$\begin{aligned} &\frac{\partial J}{\partial \sigma_j^i} = \frac{\partial J}{\partial \widehat{y}} \frac{\partial \widehat{y}}{\partial z^j} \frac{\partial z^j}{\partial \sigma_i^j} \\ &= \left(\widehat{y} - y\right) \frac{\partial\left(\frac{a}{b}\right)}{\partial z^j} \prod_{i=1}^{n} \exp\left[-\left(\frac{x_i - c_i^j}{\sigma_i^j}\right)^2\right] \left[2\frac{\left(x_i - c_i^j\right)^2}{\left(\sigma_i^j\right)^3}\right] \end{aligned} \tag{9}$$

So σ_i^j is trained by

$$\sigma_j^i\left(k+1\right) = \sigma_j^i\left(k\right) - 2\eta\left(\widehat{y} - y\right) z^i \frac{\left(w^i - \widehat{y}\right)\left(x_j - c_j^i\right)^2}{b\left(\sigma_j^i\right)^3} \tag{10}$$

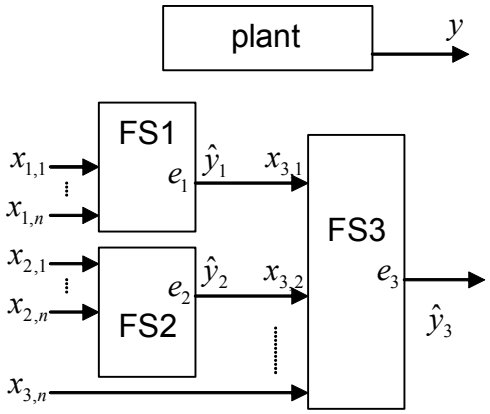

Fig. 1. A hierarchical fuzzy neural networks for identification

If we define the identification error as $e = \widehat{y}(k) - y(k)$. The gradient descent training is

$$\begin{aligned} w^i(k+1) &= w^i(k) - \eta \tfrac{z^i}{b} e \\ c_i^j(k+1) &= c_i^j(k) - 2\eta z^i(k) \frac{\left(w^i(k)-\widehat{y}(k)\right)\left(x_j(k)-c_j^i(k)\right)}{b(k)\left(\sigma_j^i(k)\right)^2} e \\ \sigma_j^i(k+1) &= \sigma_j^i(k) - 2\eta z^i(k) \frac{\left(w^i(k)-\widehat{y}(k)\right)\left(x_j(k)-c_j^i(k)\right)^2}{b(k)\left(\sigma_j^i(k)\right)^3} e \end{aligned} \tag{11}$$

The inputs and output of each subsystem are defined as in Fig.1.So the performance index is changed as $J = \frac{1}{2}\left(\widehat{y}_3 - y\right)^2$. For FS3, the learning algorithm is the same as (11), we only add subscripts in each variable, for examples, $w^i(k+1) \to w_3^i(k+1)$, $c_j^i(k+1) \to c_{3,j}^i(k+1)$, $\sigma_j^i(k+1) \to \sigma_{3,j}^i(k+1)$, $x_{3,1}(k) = \widehat{y}_1(k)$, $x_{3,2}(k) = \widehat{y}_2(k)$, $e(k) = \widehat{y}_3(k) - y(k) = e_3(k)$,

$$\widehat{y}_3 = \left(\sum_{i=1}^{l_3} w_3^i \prod_{j=1}^{n_3} \mu_{A_{3,j}^i}(x_{3,j})\right) / \left(\sum_{i=1}^{l_3} \prod_{j=1}^{n_3} \mu_{A_{3,j}^i}(x_{3,j})\right) \tag{12}$$

For subsystem FS2, if we want to update w_2^i, we should calculate

$$\frac{\partial J}{\partial w_2^i} = \frac{\partial J}{\partial \widehat{y}_3}\frac{\partial \widehat{y}_3}{\partial \widehat{y}_2}\frac{\partial \widehat{y}_2}{\partial w_2^i} \tag{13}$$

From Fig.1 we know $\frac{\partial \widehat{y}_3}{\partial \widehat{y}_2}$ corresponds to $x_{3,2}(k)$, so

$$\begin{aligned} &\tfrac{\partial J}{\partial \widehat{y}_3} = \widehat{y}_3 - y \\ &\tfrac{\partial \widehat{y}_3}{\partial \widehat{y}_2} = \tfrac{\partial \widehat{y}_3}{\partial z_3^i}\tfrac{\partial z_3^i}{\partial \widehat{y}_2} = \left[\tfrac{a_3}{b_3^2} - \tfrac{w_3^i}{b_3}\right] z_3^i \left[2\tfrac{\widehat{y}_2 - c_{3,2}^i}{\left(\sigma_{3,2}^i\right)^2}\right] \\ &\tfrac{\partial \widehat{y}_2}{\partial w_2^i} = \tfrac{z_2^i}{b_2} \end{aligned} \tag{14}$$

Because $\frac{a_3}{b_3^2} - \frac{\overline{y}_3^i}{b_3} = \frac{\widehat{y}_3 - \overline{y}_3^i}{b_3}$

$$\frac{\partial J}{\partial w_2^i} = \frac{z_2^i}{b_2} 2\frac{\widehat{y}_3 - w_3^i}{b_3} z_3^i \frac{\widehat{y}_2 - c_{3,2}^i}{\left(\sigma_{3,2}^i\right)^2} e_3(k) \tag{15}$$

$\overline{y}_2^j$ is updated by

$$w_2^i(k+1) = w_2^i(k) - \eta \frac{z_2^i}{b_2} 2\frac{\widehat{y}_3 - w_3^i}{b_3} z_3^i \frac{\widehat{y}_2 - c_{3,2}^i}{\left(\sigma_{3,2}^i\right)^2} e_3(k) \tag{16}$$

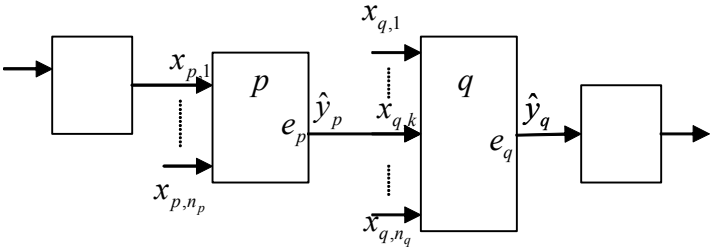

Fig. 2. Training in general case

Compare to (6), if we define

$$e_2(k) = 2\frac{\widehat{y}_3 - w_3^i}{b_3} z_3^j \frac{\widehat{y}_2 - c_{3,2}^i}{\left(\sigma_{3,2}^i\right)^2} e_3(k) \tag{17}$$

Using (14) $e_2(k) = \frac{\partial \widehat{y}_3}{\partial \widehat{y}_2} e_3(k)$. Similar we can obtain $e_1(k) = \frac{\partial \widehat{y}_3}{\partial \widehat{y}_1} e_3(k)$, where

$$\frac{\partial \widehat{y}_3}{\partial \widehat{y}_1} = 2\frac{\widehat{y}_3 - w_3^i}{b_3} z_3^j \frac{y_2 - c_{3,1}^i}{\left(\sigma_{3,1}^i\right)^2} \tag{18}$$

With $e_1(k)$ and $e_2(k)$ we can train the subsystem FS1 and FS2 independently by the normal algorithm (11). In general case shown in Fig.2, the training procedures are as follows:

1) According to the structure of the hierarchical fuzzy neural networks, we calculate the output of each sub-fuzzy neural networks by (3). Some outputs of fuzzy neural networks should be the inputs of the next level.
2) Calculate the error for each block. We start from the last block, the identification error is

$$e_o(k) = \widehat{y}_o(k) - y(k) \tag{19}$$

where $e_o(k)$ is identification error, $\widehat{y}_o(k)$ is the output of the whole hierarchical fuzzy neural networks, $y(k)$ is the output the plant. Then we back propagate the error form the structure of the hierarchical fuzzy neural networks. In Fig.2, we can calculate the error for the block p (defined as e_p) form its former block q (defined as e_q). By the chain rule discussed above

$$e_p = 2\frac{\widehat{y}_q - w_q^i}{b_q} z_q^j \frac{\widehat{y}_p - c_{q,p}^i}{\left(\sigma_{q,p}^i\right)^2} e_q \tag{20}$$

3) Train the Gaussian function (membership functions in the premise and the consequent parts) for each block independently, for $p_$th block backpropagation-like algorithm is

$$\begin{aligned} w_p^i(k+1) &= w_p^i(k) - \eta \tfrac{z_p^i}{b_p} e_p \\ c_{p,i}^j(k+1) &= c_{p,i}^j(k) - 2\eta z_p^i \frac{\left(w_p^i - \widehat{y}_p\right)\left(x_{p,j} - c_{p,j}^i\right)}{b\left(\sigma_{p,j}^i\right)^2} e_p \\ \sigma_{p.j}^i(k+1) &= \sigma_{p,j}^i(k) - 2\eta z_p^i \frac{\left(w_p^i - \widehat{y}_p\right)\left(x_{p,j} - c_{p,j}^i\right)^2}{b_p\left(\sigma_{p,j}^i\right)^3} e_p \end{aligned} \tag{21}$$

where $z_p^i = \prod_{j=1}^{n_p} \exp\left[-\left(\frac{x_{p,j} - c_{p,j}^i}{\sigma_{p,j}^i}\right)^2\right]$, $b_p = \sum_{i=1}^{l_p} z_p^i$.

III. STABLE LEARNING

If we define the identification error as

$$e(k) = \widehat{y}(k) - y(k) \tag{22}$$

By (20) $e(k)$ can be propagated to each sub-block, named $e_p(k)$, there is a virtual output $y_p(k)$ in the plant which is corresponding to the output of the sub-block $\widehat{y}_p(k)$, so

$$e_p(k) = \widehat{y}_p(k) - y_p(k) \tag{23}$$

For p_th block, we assume the nonlinear plant can be expressed in Gaussian membership function as

$$y_p = \frac{\left(\sum_{j=1}^{m} \overline{y}^{j*} \prod_{j=1}^{n} \exp\left[-\left(\frac{x_i - c_i^{j*}}{\sigma_i^{j*}}\right)^2\right]\right)}{\left(\sum_{j=1}^{m} \prod_{j=1}^{n} \exp\left[-\left(\frac{x_i - c_i^{j*}}{\sigma_i^{j*}}\right)^2\right]\right)} - \mu \tag{24}$$

where $\overline{y}^{j*}$, c_i^{j*} and σ_i^{j*} are unknown parameters which may minimize the modelling error μ. In the case of three independent variables, a smooth function f has Taylor formula as

$$\begin{aligned} f(x_1, x_2, x_3) = \sum_{k=0}^{l-1} \tfrac{1}{k!} [\left(x_1 - x_1^0\right) \tfrac{\partial}{\partial x_1} \\ + \left(x_2 - x_2^0\right) \tfrac{\partial}{\partial x_2} + \left(x_3 - x_3^0\right) \tfrac{\partial}{\partial x_3}]_0^k f + R_l \end{aligned} \tag{25}$$

where R_l is the remainder of the Taylor formula. If we let x_1, x_2, x_3 correspond $\overline{y}^j$, c_i^j and σ_i^j, x_1^0, x_2^0, x_3^0 correspond $\overline{y}^{j*}$, c_i^{j*} and σ_i^{j*},

$$\begin{aligned} \widehat{y}_p = y_p + \mu + \sum_{j=1}^{m} \left(\overline{y}^j - \overline{y}^{j*}\right) \tfrac{z^j}{b} + \sum_{j=1}^{m} \sum_{i=1}^{n} \tfrac{\partial \widehat{y}}{\partial c_i^j} \left(c_i^j - c_i^{j*}\right) \\ + \sum_{j=1}^{m} \sum_{i=1}^{n} \tfrac{\partial \widehat{y}}{\partial \sigma_i^j} \left(\sigma_i^j - \sigma_i^{j*}\right) + R_1 \end{aligned} \tag{26}$$

where R_1 is second order approximation error of the Taylor series,

$$\begin{aligned} \tfrac{\partial \widehat{y}_p}{\partial c_i^j} &= 2 z^j(k) \tfrac{\left(\overline{y}(k)^j - \widehat{y}(k)\right)\left(x_i(k) - c_i^j(k)\right)}{b(k)\left(\sigma_i^j(k)\right)^2} \\ \tfrac{\partial \widehat{y}_p}{\partial \sigma_i^j} &= 2 z^j(k) \tfrac{\left(\overline{y}(k)^j - \widehat{y}(k)\right)\left(x_i(k) - c_i^j(k)\right)^2}{b(k)\left(\sigma_i^j(k)\right)^3} \end{aligned} \tag{27}$$

So

$$\begin{aligned} \sum_{j=1}^{m} \sum_{i=1}^{n} \tfrac{\partial \widehat{y}_p}{\partial c_i^j} \left(c_i^j - c_i^{j*}\right) &= \tfrac{2}{b(k)} D_1^T(k) \widetilde{C}(k) \\ \sum_{j=1}^{m} \sum_{i=1}^{n} \tfrac{\partial \widehat{y}_p}{\partial \sigma_i^j} \left(\sigma_i^j - \sigma_i^{j*}\right) &= \tfrac{2}{b(k)} D_2^T(k) \widetilde{\Omega}(k) \end{aligned} \tag{28}$$

where

$$\widetilde{C}(k) = \begin{bmatrix} \left(c_1^1 - c_1^{1*}\right) \cdots \left(c_n^1 - c_n^{1*}\right) \\ \cdots \left(c_1^m - c_1^{m*}\right) \cdots \left(c_n^m - c_n^{m*}\right) \end{bmatrix}^T \in R^{n+m}$$

$$D_1(k) = \begin{bmatrix} z^1(k) \tfrac{\left(\overline{y}(k)^1 - \widehat{y}(k)\right)\left(x_1(k) - c_1^1(k)\right)}{\left(\sigma_1^1(k)\right)^2} \\ \cdots z^1(k) \tfrac{\left(\overline{y}(k)^1 - \widehat{y}(k)\right)\left(x_n(k) - c_n^1(k)\right)}{\left(\sigma_n^1(k)\right)^2} \\ \cdots z^m(k) \tfrac{\left(\overline{y}(k)^m - \widehat{y}(k)\right)\left(x_1(k) - c_1^m(k)\right)}{\left(\sigma_1^m(k)\right)^2} \\ \cdots z^m(k) \tfrac{\left(\overline{y}(k)^m - \widehat{y}(k)\right)\left(x_n(k) - c_n^m(k)\right)}{\left(\sigma_n^1(k)\right)^2} \end{bmatrix}^T$$

$$\widetilde{\Omega}(k) = \begin{bmatrix} \left(\sigma_1^1 - \sigma_1^{1*}\right) \cdots \left(\sigma_n^1 - \sigma_n^{1*}\right) \\ \cdots \left(\sigma_1^m - \sigma_1^{m*}\right) \cdots \left(\sigma_n^m - \sigma_n^{m*}\right) \end{bmatrix}^T \in R^{n+m}$$

$$D_2(k) = \begin{bmatrix} z^1(k) \tfrac{\left(\overline{y}(k)^1 - \widehat{y}(k)\right)\left(x_1(k) - c_1^1(k)\right)^2}{\left(\sigma_1^1(k)\right)^3} \\ \cdots z^1(k) \tfrac{\left(\overline{y}(k)^1 - \widehat{y}(k)\right)\left(x_n(k) - c_n^1(k)\right)^2}{\left(\sigma_n^1(k)\right)^3} \\ \cdots z^m(k) \tfrac{\left(\overline{y}(k)^m - \widehat{y}(k)\right)(x_1(k) - c_1^m(k))^2}{\left(\sigma_1^m(k)\right)^3} \\ \cdots z^m(k) \tfrac{\left(\overline{y}(k)^m - \widehat{y}(k)\right)(x_n(k) - c_n^m(k))^2}{\left(\sigma_n^m(k)\right)^3} \end{bmatrix}^T \tag{29}$$

$$\begin{aligned} e_p(k) = \tfrac{1}{b(k)} z^T(k) \widetilde{y}(k) + \tfrac{2}{b(k)} D_1^T(k) \widetilde{C}(k) \\ + \tfrac{2}{b(k)} D_2^T(k) \widetilde{\Omega}(k) + \zeta(k) \end{aligned} \tag{30}$$

where $\widetilde{y}_k = \overline{y}(k) - \overline{y}^*(k)$, $\overline{y}(k) = \left[\overline{y}^1, \cdots, \overline{y}^m\right]^T$, $z(k) = \left[z^1 \cdots z^m\right]^T$, $\zeta(k) = R_1 + \mu$.

Theorem 1: If we use Mamdani-type fuzzy neural network (3) to identify nonlinear plant (1), the following backpropagation algorithm makes identification error $e_p(k)$ bounded

$$\begin{aligned} \overline{y}(k+1) &= \overline{y}(k+1) - \tfrac{\eta(k)}{b(k)} z(k) e_p(k) \\ C(k+1) &= C(k) - 2\tfrac{\eta(k)}{b(k)} D_1(k) e_p(k) \\ \Omega(k+1) &= \Omega(k) - 2\tfrac{\eta(k)}{b(k)} D_2(k) e_p(k) \end{aligned} \tag{31}$$

$\eta(k) = \frac{\eta}{1+\Phi(k)^2}$, $\Phi(k) = \|z(k)\|^2 + 4\|D_1(k)\|^2 + 4\|D_2(k)\|^2$, $0 < \eta \le \max_k \left\{b^2(k)\right\}$. The average of the identification error satisfies

$$J = \limsup_{T \to \infty} \frac{1}{T} \sum_{k=1}^{T} e_p^2(k) \le \frac{1}{\pi} \overline{\zeta} \tag{32}$$

where $\pi = \frac{\eta}{(1+\Phi(k))^2} > 0$, $\overline{\zeta} = \max_k \left[\zeta^2(k)\right]$

Proof: We selected a positive defined scalar L_k as

$$L_k = \|\widetilde{y}(k)\|^2 + \left\|\widetilde{C}(k)\right\|^2 + \left\|\widetilde{\Omega}(k)\right\|^2 \tag{33}$$

The updating law (31) can be written as

$$\begin{aligned} \widetilde{y}(k+1) &= \widetilde{y}(k) - \tfrac{\eta(k)}{b(k)} z(k) e(k) \\ \widetilde{C}(k+1) &= \widetilde{C}(k) - 2\tfrac{\eta(k)}{b(k)} D_1(k) e(k) \\ \widetilde{\Omega}(k+1) &= \widetilde{\Omega}(k) - 2\tfrac{\eta(k)}{b(k)} D_2(k) e(k) \end{aligned} \tag{34}$$

So we have

$$\begin{aligned}
&\Delta L_k = \left\|\widetilde{y}(k) - \tfrac{\eta(k)}{b(k)} z(k)\, e(k)\right\|^2 - \|\widetilde{y}(k)\|^2 \\
&+ \left\|\widetilde{C}(k) - 2\tfrac{\eta(k)}{b(k)} D_1(k)\, e(k)\right\|^2 - \left\|\widetilde{C}(k)\right\|^2 \\
&+ \left\|\widetilde{\Omega}(k) - 2\tfrac{\eta(k)}{b(k)} D_2(k)\, e(k)\right\|^2 - \left\|\widetilde{\Omega}(k)\right\|^2 \\
&= \eta^2(k) \left\|\tfrac{z(k)}{b(k)}\right\|^2 e^2(k) - 2\eta(k)\, \tfrac{z^T(k)\widetilde{y}(k)}{b(k)} e(k) \\
&+4\eta^2(k) \left\|\tfrac{D_1(k)}{b(k)}\right\|^2 e^2(k) - 4\eta(k)\, \tfrac{D_1^T(k)\widetilde{C}(k)}{b(k)} e(k) \\
&+4\eta^2(k) \left\|\tfrac{D_2(k)}{b(k)}\right\|^2 e^2(k) - 4\eta(k)\, \tfrac{D_1^T(k)\widetilde{\Omega}(k)}{b(k)} e(k) \\
&= \tfrac{\eta^2(k)}{b^2(k)} e^2(k) \left(\|z(k)\|^2 + 4\|D_1(k)\|^2 + 4\|D_2(k)\|^2\right) \\
&-2\eta(k)\, e(k) [\tfrac{1}{b(k)} z^T(k)\, \widetilde{y}(k) + \tfrac{2}{b(k)} D_1^T(k)\, \widetilde{C}(k) \\
&+\tfrac{2}{b(k)} D_1^T(k)\, \widetilde{\Omega}(k)]
\end{aligned} \tag{35}$$

Because $e(k) = \frac{1}{b(k)} z^T(k)\, \widetilde{y}(k) + \frac{2}{b(k)} D_1^T(k)\, \widetilde{C}(k) + \frac{2}{b(k)} D_2^T(k)\, \widetilde{\Omega}(k) + \zeta(k)$, the last term of (35) is $-2\eta(k)\, e(k)\, [e(k) - \zeta(k)]$. Because $\eta_k > 0$

$$\begin{aligned}
&-2\eta(k)\, e(k)\, [e(k) - \zeta(k)] \\
&= -2\eta(k)\, e^2(k) + 2\eta(k)\, e(k)\, \zeta(k) \\
&\le -2\eta(k)\, e^2(k) + \eta(k)\, e^2(k) + \eta(k)\, \zeta^2(k) \\
&= -\eta(k)\, e^2(k) + \eta(k)\, \zeta^2(k)
\end{aligned} \tag{36}$$

So

$$\begin{aligned}
&\Delta L_k \le \tfrac{\eta^2(k)}{b^2(k)} e^2(k) \left(\|z(k)\|^2 + 4\|D_1(k)\|^2 + 4\|D_2(k)\|^2\right) \\
&-\eta(k)\, e^2(k) + \eta(k)\, \zeta^2(k) \\
&= -\eta(k)\, e^2(k) [1 - \tfrac{\eta(k)}{b^2(k)} (\|z(k)\|^2 \\
&+4\|D_1(k)\|^2 + 4\|D_2(k)\|^2)] + \eta_k \zeta_k^2
\end{aligned} \tag{37}$$

We define $\Phi(k) = \|z(k)\|^2 + 4\|D_1(k)\|^2 + 4\|D_2(k)\|^2$, and we choose $\eta(k)$ as $\eta(k) = \frac{\eta}{1+\Phi_k} \le 1$. Becasue $\eta \le \max_k \{b(k)\}$, $\frac{\eta}{b^2(k)} \le \frac{\max_k \{b^2(k)\}}{b^2(k)} \le 1$

$$\begin{aligned}
&\Delta L_k \le -\tfrac{\eta}{1+\Phi(k)} e^2(k) \left[1 - \tfrac{\eta}{b^2(k)} \tfrac{\Phi(k)}{1+\Phi(k)}\right] \\
&+\eta(k)\, \zeta^2(k) \\
&\le -\tfrac{\eta}{1+\Phi(k)} e^2(k) \left[1 - \tfrac{1}{1+\Phi(k)} \Phi(k)\right] \\
&+\eta(k)\, \zeta^2(k) \\
&\le -\pi e^2(k) + \zeta^2(k)
\end{aligned} \tag{38}$$

where π is defined in (32). Because$\|\widetilde{y}(k)\|^2 + \left\|\widetilde{C}(k)\right\|^2 + \left\|\widetilde{\Omega}(k)\right\|^2$

$$\begin{aligned}
&n[\min(\widetilde{y}(k)) + \min(\widetilde{c}(k)) + \min(\widetilde{\sigma}(k))] \\
&\le L_k \le n[\max(\widetilde{y}(k)) + \max(\widetilde{c}(k)) + \max(\widetilde{\sigma}(k))]
\end{aligned} \tag{39}$$

where

$$\begin{aligned}
&n[\min(\widetilde{y}(k)) + \min(\widetilde{c}(k)) + \min(\widetilde{\sigma}(k))] \\
&n[\max(\widetilde{y}(k)) + \max(\widetilde{c}(k)) + \max(\widetilde{\sigma}(k))]
\end{aligned} \tag{40}$$

are $\mathcal{K}_\infty$-functions, and πe_k^2 is an $\mathcal{K}_\infty$-function, $\zeta^2(k)$ is a $\mathcal{K}$-function. From (33) we know L_k is the function of $e(k)$ and ζ_k, so L_k admits a smooth ISS-Lyapunov function as in *Definition 2*. From *Theorem 1*, the dynamic of the identification error is input-to-state stable. Because the "INPUT" ζ_k is bounded and the dynamic is ISS, the "STATE" e_k is bounded.

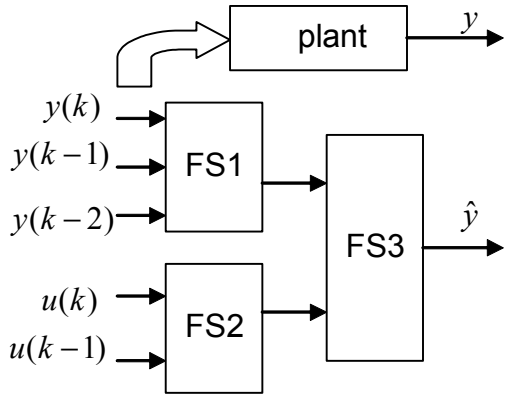

Fig. 3. Hierarchical fuzzy neural to identify a nonlinear system

(38) can be rewritten as

$$\Delta L_k \le -\pi e_k^2 + \eta \zeta_k^2 \le \pi e_k^2 + \overline{\zeta} \tag{41}$$

where $\overline{\zeta} = \max_k \left[\zeta_k^2\right]$. Summarizing (41) from 1 up to T, and by using $L_T > 0$ and L_1 is a constant, we obtain

$$\begin{aligned}
L_T - L_1 &\le -\pi \textstyle\sum_{k=1}^T e_k^2 + T\eta\overline{\zeta} \\
\pi \textstyle\sum_{k=1}^T e_k^2 &\le L_1 - L_T + T\eta\overline{\zeta} \le L_1 + T\eta\overline{\zeta}
\end{aligned} \tag{42}$$

(32) is established. ■

IV. SIMULATIONS

We will use the nonlinear system which proposed [11] and [14] to illustrate the training algorithm for hierarchical fuzzy neural networks. The identified nonlinear plant is

$$y(k+1) = \tfrac{y(k)y(k-1)y(k-2)u(k-1)[y(k-2)-1]+u(k)}{1+y(k-1)^2+y(k-2)^2}$$

The input vector is

$$X(k) = [y(k), y(k-1), y(k-2), u(k), u(k-1)] \tag{43}$$

The unknown nonlinear system has the standard form

$$y(k+1) = f[X(k)] \tag{44}$$

We use the following hierarchical fuzzy neural networks to identify it, see Fig.3.

We use 2 rules for each block, $l_1 = l_2 = l_3 = 2$. The input numbers for each input are $n_1 = 3$, $n_2 = 2$, $n_3 = 2$. We use 200 data to train the model, the training input is used as in [14]. The identification results are shown in Fig.4.

Now we compare our algorithm with normal fuzzy neural networks [1][6][8], here we use 20 rules. The training rule is (11). Let us define the mean squared error for finite time is $J(k) = \frac{1}{2k} \sum_{i=1}^k e^2(i)$. The comparison results are shown in Fig. 5. We can see that compared to normal fuzzy neural networks, hierarchical fuzzy neural networks can model nonlinear system with less rules. By the training

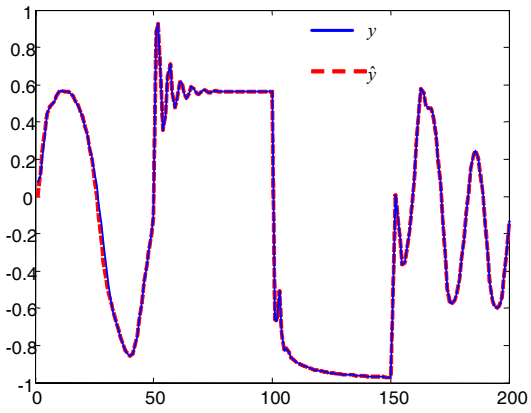

Fig. 4. Identification with hierarchical fuzzy neural networks

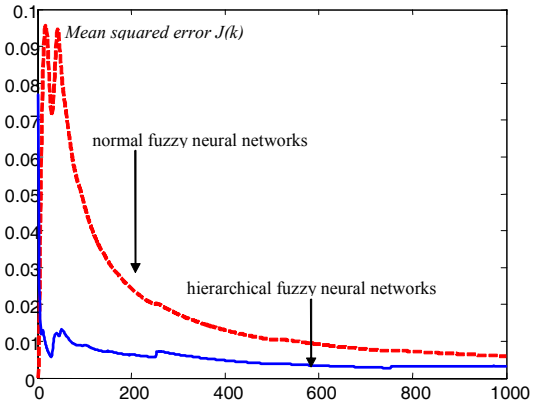

Fig. 5. Comparison

algorithm proposed in this paper, the convergence speed is faster than the normal one.

V. CONCLUSIONS

In this paper we propose a simple training algorithm for hierarchical fuzzy neural networks. The modelling process can be realized in each sub-block independently. Further works will be done on structure training and adaptive control. The new stable algorithms with time-varying learning rates are applied to hierarchical fuzzy neural networks.

REFERENCES

[1] M.Brown, C.J.Harris, *Neurofuzzy Aadaptive Modelling and Control*, Prentice Hall: New York , 1994.

[2] M.Y.Chen and D.A.Linkensm, A systematic neuro-fuzzy modeling framework with application to material property prediction, *IEEE Trans. Syst., Man, Cybern. B*, Vol.31, 781-790, 2001.

[3] D.S.Chen and R.C.Jain, A robust back propagation learning algorithm for function approximation, *IEEE Trans. Neural Networks*, Vol.5, No.3, 1994.

[4] F.L.Chung, J.C.Duan, On Multistage Fuzzy Neural Network Modeling, *IEEE Transactions on Fuzzy Systems*, Vol.8, No.2, 125-142, 2000.

[5] P.A.Ioannou and J.Sun, *Robust Adaptive Control*, Prentice-Hall, Inc, Upper Saddle River: NJ, 1996.

[6] J. S. Juang, ANFIS: Adaptive-network-based fuzzy inference system, *IEEE Transactions on Systems, Man and Cybernetics,* vol. 23, 665-685, 1993.

[7] M.L. Lee, H.Y. Chung and F.M. Yu, Modeling of hierarchical fuzzy systems, *Fuzzy sets and systems*, 138, 343-361, 2003.

[8] C.T.Lin and G.Lee, *Neural fuzzy systems: A neural-fuzzy synergism to intelligent systems*, Prentice-Hall Inc., NJ, 1996.

[9] C.T.Lin, A neural fuzzy control system with structure and parameter learning, *Fuzzy Sets and Systems.*, Vol.70, 183-212, 1995.

[10] Y.G.Leu, T.T.Lee and W.Y.Wang, Observer-based adaptive fuzzy-neural control for unknown nonlinear dynamical systems, *IEEE Trans. Syst., Man, Cybern. B*, Vol.29, 583-591, 1999.

[11] K.S.Narendra and K.Parthasarathy, Identification and Control of Dynamical Systems Using Neural Networks, *IEEE Trans. Neural Networks*, Vol.1, No.1, 4-27, 1990.

[12] G. V. S. Raju, J. Zhou and R. A. Kisner, Hierarchical fuzzy control, *Int. J. of Control*, 54, no. 5, pp. 1201-1216, 1991

[13] W. Rattasiri and S.K. Halgamuge, Computationally Advantageous and Stable Hierarchical Fuzzy Systems for Active Suspension, *IEEE Trans. on Industrial Electronics*, Vol. 50, No. 1, 48-61, 2003.

[14] P. S. Sastry, G. Santharam, and K. P. Unnikrishnan, Memory neural networks for identification and control of dynamic systems," *IEEE Trans. Neural Networks*, vol. 5, 306-319, 1994.

[15] L. X. Wang, *A course in fuzzy systems and control*, Prentince Hall Inc., 1997.

[16] L. X. Wang, Analysis and Design of Hierarchical Fuzzy Systems, *IEEE Transactions on Fuzzy Systems*, Vol.7, No.3, 617-624, 1999.

[17] C. Wei and L.X. Wang, A Note on Universal Approximation by Hierarchical Fuzzy Systems, *Information Sciences*, Vol. 123, 241-248, 2000.

[18] S.Wu and M.J.Er, Dynamic fuzzy neural networks- a novel approach to function approximation, *IEEE Trans. Syst., Man, Cybern. B*, Vol.30, 358-364, 2000.

[19] R.R. Yager, "On the construction of Hierarchical Fuzzy Systems Models, *IEEE Trans. Syst., Man, Cybern. C*, Vol. 28, 55-66, 1998.

[20] W.Yu and X. Li, Some stability properties of dynamic neural networks, *IEEE Trans. Circuits and Systems, Part I*, Vol.48, No.1, 256-259, 2001.

[21] W.Yu and X. Li, Some new results on system identification with dynamic neural networks, *IEEE Trans. Neural Networks*, Vol.12, No.2, 412-417, 2001.

[22] Wen Yu, Xiaoou Li, Fuzzy identification using fuzzy neural networks with stable learning algorithms, *IEEE Transactions on Fuzzy Systems*, Vol.12, No.3, 411-420, 2004.