A New Learning Algorithm for a Fully Connected Neuro-Fuzzy Inference System

C. L. Philip Chen, *Fellow, IEEE*, Jing Wang, *Student Member, IEEE*, Chi-Hsu Wang, *Fellow, IEEE*, and Long Chen, *Member, IEEE*

***Abstract*— A traditional neuro-fuzzy system is transformed into an equivalent fully connected three layer neural network (NN), namely, the fully connected neuro-fuzzy inference systems (F-CONFIS). The F-CONFIS differs from traditional NNs by its dependent and repeated weights between input and hidden layers and can be considered as the variation of a kind of multilayer NN. Therefore, an efficient learning algorithm for the F-CONFIS to cope these repeated weights is derived. Furthermore, a dynamic learning rate is proposed for neuro-fuzzy systems via F-CONFIS where both premise (hidden) and consequent portions are considered. Several simulation results indicate that the proposed approach achieves much better accuracy and fast convergence.**

***Index Terms*— Fully connected neuro-fuzzy inference systems (F-CONFIS), fuzzy logic, fuzzy neural networks, gradient descent, neural networks (NNs), neuro-fuzzy system, optimal learning.**

I. Introduction

NEURO-FUZZY systems have been applied to many engineering applications in decades related to pattern recognition, intelligent adaptive control, regression and density estimation, systems modeling, and so on [1]–[6]. A neuro-fuzzy system possesses characteristics of neural network (NN), linguistic description and logic control [7]–[10]. Although the significant progress has been made by combining different learning algorithms with neuro-fuzzy system [11]–[16], there are still problems that need to be solved for practical implementations for instance, finding the optimal learning rates for both the premise and consequent parts to increase convergence speed, or updating the parameters of membership functions (MFs). In a neuro-fuzzy system, in general, the rule layer is a product layer instead of a summing layer in a conventional feedforward NN. As a result, it is not concise to apply learning algorithms in turning premise parameters. Therefore, to design a systematic learning for the neuro-fuzzy system, a traditional neuro-fuzzy system is reformulated as an equivalent fully connected three-layer NN, i.e., the fully connected fuzzy inference systems (F-CONFIS) [17]. Though some literatures have proved the functional equivalence between a fuzzy system and a NN, they are nonconstructive [9]. The F-CONFIS provides constructive steps to build the equivalence between a neuro-fuzzy system and NN. The F-CONFIS is different with the classical multiple layer NNs by its repeated link weights. With some special arrangements, we can derive the training algorithm for the F-CONFIS, thereafter for a neuro-fuzzy system efficiently and effectively.

Choosing a proper learning rate is a critical issue in the gradient descent algorithm for a neuro-fuzzy system. It is a time consuming process to select manually and the result may lack of generality. In a neuro-fuzzy system, it needs to adjust two learning rates corresponding to adjustable parameters in the premise and the consequent parts. To improve the convergence rate, although many kinds of dynamic learning rate methods have been proposed for neuro-fuzzy systems [16], [18]–[22], the convergence rate was still not improved satisfactorily because only the optimal learning rate of the consequent part is derived. Although the learning rate of the premise part can be obtained by genetic search techniques in [18], of which is time consuming and the optimal result is not guaranteed. The optimal learning rate of the consequent part can be obtained analytically, but unfortunately, the analytical expression of the premise part cannot be solved in a similar way because the premise part is usually a transcendental equation. Due to the complexity of the premise part of neuro-fuzzy systems, there is no existing learning algorithm about the optimal learning rate of the premise portion that has been discussed. In this paper, by transforming a neuro-fuzzy system to F-CONFIS, a dynamic learning algorithm is proposed.

The major contributions of the paper are summarized as follows.

1) We derive the ordinal indices for hidden neurons of the repeated weights and, thereafter, the gradient descent-based training algorithm of F-CONFIS.
2) The dynamic optimal learning rates of neuro-fuzzy system are derived first time not only in the consequent part, but also in the premise part. The optimal learning rate for the parameters of MFs can greatly boost the training speed of the neuro-fuzzy systems. Such kind of speedup is extremely valuable for the online applications.
3) In addition, this paper explicitly provides formulations to update the parameters of fuzzy MFs in the premise part

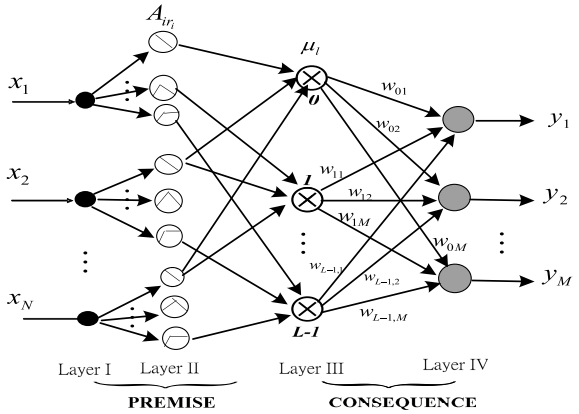
Manuscript received August 27, 2013; accepted February 5, 2014. Date of publication April 24, 2014; date of current version September 15, 2014. This work was supported in part by the National 973 Basic Research Program of China under Grant 2011CB302801, in part by the Multiyear Research Grants, University of Macau, and in part by the Macau Science and Technology Development under Grant 008/2010/A1.

C. L. P. Chen, J. Wang, and L. Chen are with the Faculty of Science and Technology, University of Macau, Macau 99999, China (e-mail: philip.chen@ieee.org; elizawangj@gmail.com; longchen@umac.mo).

C.-H. Wang is with the Department of Electrical and Computer Engineering, National Chiao Tung University, Hsinchu 300, Taiwan, and also with North East University, Qinhuangdao 066004, China (e-mail: cwang@cn.nctu.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2014.2306915



$\{x_i, i = 1, 2, 3, \mathrm{L}, N\}$: the input vector

$\{y_k, k = 1, 2, \ldots, M\}$: the output vector

R_i : the total number of MFs for fuzzy variable x_i

L : the total number of fuzzy rules

l : the rule number. $(0 \le l \le L-1)$

r_i : the ordinal index of r th MFs for fuzzy variable x_i $(0 \le r_i \le R_i - 1)$

$A_{ir_i}(x_i)$: the r_i th MFfor fuzzy variable x_i

W_{lk}: the weight of the consequent part

Fig. 1. Common configuration of neuro-fuzzy system model.

and the weights of the consequent part of a neuro-fuzzy system via its equivalent F-CONFIS. This simplifies the implementation of the optimal learning of fuzzy inference systems.

This paper is organized as follows. The special properties of F-CONFIS are discussed in Section II. The new updating laws of the gradient descent algorithm and a new optimal learning with special properties for F-CONFIS are proposed in Section III. The discussion of the time complexity and derivative-based method are given in Section IV. The simulation results and the comparison analysis are illustrated in Section V. Finally, this paper is concluded in Section VI.

II. Special Properties of Equivalent F-CONIS Systems

A. Fully Connected Neuro-Fuzzy Inference Systems

The equality between fuzzy logic systems and the feedforward NNs have been proved in [9]. The specific form of an equivalent fully connected three layer feedforward NN for the neuro-fuzzy system was discussed in [17]. Fig. 1 shows the common configuration of a neuro-fuzzy system model [18], [20]. It is comprised of four layers. The first layer is the input layer, whose nodes correspond to input variables. Layer II is the MF layer, in which the value of a node quantifies the degree of MF of the input linguist variable. Layer III is the fuzzy rule layer. Therein each node represents a fuzzy rule. The last layer is the output layer. In this configuration, the consequent part is a fully connected graph, but apparently the premise part is not, because the nodes in the MF layer do not connect to all nodes in the fuzzy rule layer.

The fuzzy Mamdani model [23] represented by above configuration of a neuro-fuzzy system contains following rules.

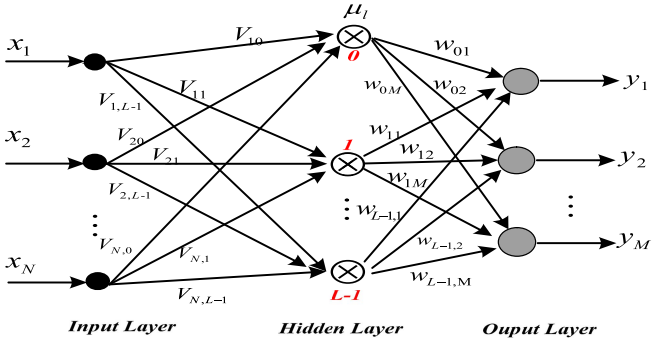


Fig. 2. Complete F-CONFIS.

Rule l: IF x_1 is A_{1r_1} and ... and x_N is A_{Nr_N} THEN y_1 is B_{l1} and .... and y_M is B_{lM} where $A_{1r_1}, \ldots, A_{Nr_N}$ and $B_{l1}, \ldots, B_{lM}$ are the standard fuzzy sets defined by MFs. When the input variables are given, the firing strength μ_l of the lth rule is defined by

$$\mu_l = A_{1r_1}(x_1) \times \cdots \times A_{Nr_N}(x_N) \quad \{l = 0, 1, 2, \ldots, L-1; r_i = 0, 1, 2, \ldots, R_i - 1\}. \quad (1)$$

There are two difficulties in developing a learning algorithm for the neuro-fuzzy system. One is that, in a neuro-fuzzy system, the links between the MF layer, and the fuzzy rule layer are not fully connected, so it is selective learning. The other is that the operators in the fuzzy rule layer are product-form rather than summation form.

This kind of four-layer neuro-fuzzy system can be transformed into an equivalent F-CONFIS [17]. For convenience and completeness, we briefly describe the derivation of F-CONFIS here. In order to overcome the above mentioned difficulties, the MF layer II of a neuro-fuzzy system in Fig. 1 can be redrawn, and substituted for new links between Layer I and Layer III, so that we can have a fully connected three-layer NN as shown in Fig. 2. The new link weight $\{V_{ij}(i = 1, 2, \ldots, n; j = 1, 2, \ldots, L)\}$ in Fig. 2 is represented as between the ith node of input Layer and the jth node of the fuzzy rule layer. The exponential function was taken as the activation function in F-CONFIS. Therefore, the new link weight $V_{ij}(i = 1, 2, \ldots, N; j = 0, 1, \ldots, L-1)$ is defined as

$$V_{ij} = \ln(A_{ir_i(j)}(x_i)) \quad (2)$$

where ln is the natural logarithm, and $r_i(j)$ $(i = 1, \ldots, N)$ are ordinal indices of MFs corresponding to rule j.

Fig. 2 depicts the equivalent F-CONFIS, and it is equivalent to the original neuro-fuzzy system shown in Fig. 1. The F-CONFIS has three layers as shown in Fig. 2 and its hidden layer is also the fuzzy rule layer.

B. Special Properties of F-CONFIS

The F-CONFIS differs from traditional multilayer NNs. For a normal multilayer feedforward NN, every weight is updated only once to reduce the error in the process of each epoch. While in F-CONFIS, the weights may be updated

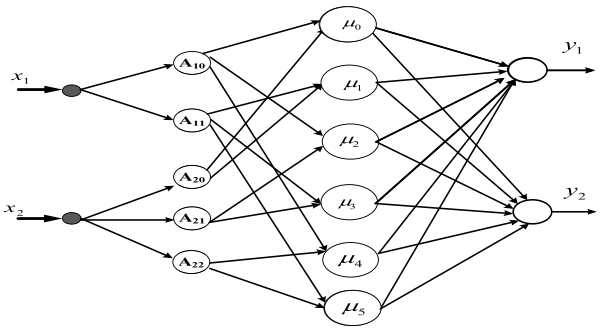


Fig. 3. Neuro-fuzzy system with two input variables and two output variables.

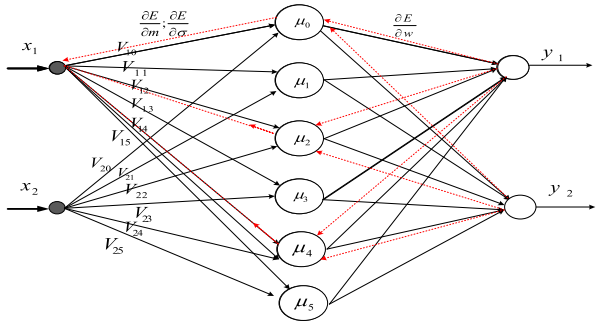


Fig. 4. Red dotted lines: gradient decent flow of F-CONFIS for the first MF of $x1$ with repeated link weight.

TABLE I
F-CONFIS WEIGHTS GENERATED FROM MFS AND RULE NUMBER

$l(=j)$	$r_1(l)$	$r_2(l)$	V_{1j}	V_{2j}
0	0	0	$ln(A_{10}(x_1))$	$ln(A_{20}(x_2))$
1	1	0	$ln(A_{11}(x_1))$	$ln(A_{20}(x_2))$
2	0	1	$ln(A_{10}(x_1))$	$ln(A_{21}(x_2))$
3	1	1	$ln(A_{11}(x_1))$	$ln(A_{21}(x_2))$
4	0	2	$ln(A_{10}(x_1))$	$ln(A_{22}(x_2))$
5	1	2	$ln(A_{11}(x_1))$	$ln(A_{22}(x_2))$

more than one time. For example, see the configuration of a neuro-fuzzy system in Fig. 3 and its equivalent F-CONFIS shown in Fig. 4. The configuration of the neuro-fuzzy system (or the F-CONFIS) has two input variables x_1 and x_2, two output variables y_1 and y_2, and two MFs $\{A_{10}(x_1), A_{11}(x_1)\}$ for x_1 and three MFs $\{A_{20}(x_2), A_{21}(x_2), A_{22}(x_2)\}$ for x_2, $\{R_1 = 2, R_2 = 3\}$. For instance, in Fig. 4, the links V_{10}, V_{12}, and V_{14} are with the same weight of $\ln(A_{10}(x_1))$ from the first MF of x_1 to hidden neurons as shown in Table I. This kind of multiple updates should be handled carefully and will be discussed next.

With above description, it is important to find the number of repeated links RW(i) for each fuzzy variable x_i in premise part so that the training algorithm can be properly carried out for the F-CONFIS. The following proposition 1 shows a precise formula of finding the number of repeated links RW(i) in the F-CONFIS.

Proposition 1: In a F-CONFIS, each input variable x_i has the number of repeated links RW(i), every same value of weights $A_{ir_i}(x_i)$ will be repeated for RW(i) times between input layer and hidden layer, RW(i) is given as

$$\mathrm{RW}(i) = \frac{\prod_{i=1}^{n} R_i}{R_i} (i = 1, \ldots, N) \tag{3}$$

where R_i is the number of MFs for input variable x_i, $A_{ir_i}(x_i)$ is the r_ith MF of x_i.

From (1), we have

$$\mu_l = \prod_{i=1}^{n} A_{ir_i}(x_i) \quad \{l = 0, 1, 2, \ldots, L-1; r_i = 0, 1, 2, \ldots, R_i - 1\}. \tag{4}$$

To be identical with l defined in [24], the ordinal index l of the lth fuzzy rule is

$$l = r_1 + \sum_{i=2}^{N} \left(\prod_{j=1}^{i-1} R_j \right) r_i. \tag{5}$$

From (4) and (5), the total number of fuzzy rules is

$$L = R_1 \times R_2 \times \cdots \times R_i \times \cdots \times R_N. \tag{6}$$

In F-CONFIS, there are L links from every node of the input layer to all of nodes of the fuzzy rule layer. However, there are only R_i different MFs for $x_i\{A_{iq} | q = 0, 1, 2, \ldots, R_i - 1\}$. From (4) and (6), it is obvious that for each fuzzy variable x_i to have the number of repeated links, RW(i), with the same weights

$$\mathrm{RW}(i) = \frac{R_1 \times R_2 \times \cdots \times R_i \times \cdots \times R_n}{R_i} = \frac{\prod_{i=1}^{n} R_i}{R_i}.$$

For instance, in Fig. 4, the number of repeated links of variables x_2 is as follows:

$$\mathrm{RW}(2) = \frac{\prod_{i=1}^{n} R_i}{R_2} = \frac{R_1 * R_2}{R_2} = \frac{6}{3} = 2$$

which are $V_{20} = V_{21} = \ln(A_{20}(x_2))$, $V_{22} = V_{23} = \ln(A_{21}(x_2))$, and $V_{24} = V_{25} = \ln(A_{22}(x_2))$ for fuzzy variable x_2 as shown in Table I.

The derivation of a new learning algorithm for F-CONFIS is discussed in the following sections.

III. NEW OPTIMAL LEARNING OF F-CONFIS

The F-CONFIS is a new type of NN that has dependent and repeated links between input and hidden layers. With Proposition 1 that shows a precise formula of finding the number of repeated links, RW(i), in F-CONFIS, an explicit learning algorithm considering the dependent and repeated weights is proposed next.

A gradient decent algorithm, may be the most commonly used, with fixed learning rates may lead to slow convergence [25]–[30]. The choosing of proper learning rate becomes a critical issue in a gradient decent-based algorithm. To deal with

this issue, many kinds of dynamic learning rate methods have been proposed [12]–[16]. However, only the optimal learning rate of the consequent part is derived for neuro-fuzzy systems [18]–[22]. Although the learning rate for the premise part is obtained by genetic search techniques [18], it is generally time consuming and the searched result may not be optimal. Due to the complexity of the premise part of neuro-fuzzy systems, no existing learning algorithm about the optimal learning of the premise portion has been discussed. In this section, based on the F-CONFIS, we derive the new optimal learning rate for both the premise part and the consequent part.

A popular centroid defuzzification method is used in the output layer as

$$y_k = \sum_{j=0}^{L-1} \mu_j w_{jk} \Big/ \sum_{j=0}^{L-1} \mu_j, (k = 1, 2, \ldots, M) \tag{7}$$

where the output vectors is $\{y_k, k = 1, 2, \ldots, M\}$. The number of training patterns is P. The total training error [17] is given as

$$E(W) = \frac{1}{2P} \sum_{p=1}^{P} \sum_{k=1}^{M} \left(y_k^p - d_k^p\right)^2 \tag{8}$$

where y_k^p is the kth actual output and d_k^p is the kth desired output. In F-CONFIS, the Gaussian MFs are adopted

$$A_{iq}(x) = \exp\left[\frac{-(x - m_{iq})^2}{2\sigma_{iq}^2}\right] \quad i = 1, \ldots, N;\ q = 0, \ldots, R_i - 1_i \tag{9}$$

where m_{iq} and σ_{iq} denote the center and width of the MFs, respectively.

In F-CONFIS and its corresponding neuro-fuzzy system, two types of adjustable parameters should be adjusted, i.e., the adjustable parameters, W consist of the parameters of MFs in the premise part and weight factors in consequent part. Therefore, weighting vector W is defined as

$$W = [m_{iq}\ \sigma_{iq}\ w_{jk}] \quad i = 1, \ldots, N;\ q = 0, \ldots, R_i - 1;\ k = 1, \ldots M \tag{10}$$

where w_{jk} are the link weights between the fuzzy rule layer and output layer. The gradient of $E(W)$ with respect to W is

$$g = \left[\frac{\partial E}{\partial m_{iq}}\ \frac{\partial E}{\partial \sigma_{iq}}\ \frac{\partial E}{\partial w_{jk}}\right] \quad i = 1, \ldots, N;\ q = 0, \ldots, R_i - 1 \quad j = 0, \ldots, L - 1;\ k = 1, \ldots, M. \tag{11}$$

The weighting vector W can be tuned as

$$W(\alpha, \beta) = W - \left[\alpha \frac{\partial E}{\partial m_{iq}}, \alpha \frac{\partial E}{\partial \sigma_{iq}}, \beta \frac{\partial E}{\partial w_{jk}}\right] \quad i = 1, \ldots, N;\ q = 0, \ldots, R_i - 1 \quad j = 0, \ldots, L - 1;\ k = 1, \ldots, M. \tag{12}$$

From (5), all the $r_i's$ for a specific rule number l are represented as

$$r_1(l) = l\%R_1; r_i(l) = \left(l / \prod_{k=1}^{i-1} R_k\right)\%R_i,\ i = 2, \ldots, N. \tag{13}$$

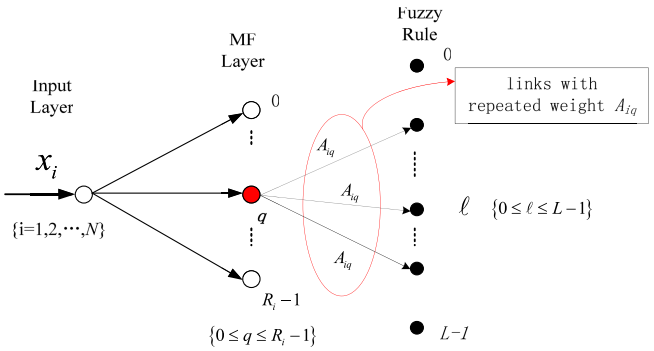


Fig. 5. Finding the links with repeated weights.

From Fig. 5, we know that the repeated weight for the qth MF of x_i is equal to $A_{iq} = A_{ir_i(\ell)}$. Therefore, we have

$$q = r_i(\ell_\lambda) \Rightarrow \{\ell_\lambda = h_\lambda(i, q);\ \lambda = 1, \ldots, \mathrm{RW}(i)\} \tag{14}$$

where $\{\ell_\lambda = h_\lambda(i, q);\ \lambda = 1, \ldots, \mathrm{RW}(i)\}$ in (14) are the links with repeated weight A_{iq} for the qth MF of x_i. For convenience, we define following s_i:

$$\begin{cases} s_i = 1, & i = 1 \\ s_i = \prod_{k=1}^{i-1} R_k, & i > 1 \end{cases} \quad \text{or} \quad \begin{cases} s_1 = 1 \\ s_{i+1} = s_i R_i, i = 1, \ldots, N - 1 \end{cases} \tag{15}$$

so that we can rewrite (13) to find the lth fuzzy rule in hidden layer as

$$l(r_1, r_2, \ldots, r_N) = \sum_{n=1}^{N} r_n s_n. \tag{16}$$

Let $A = \{r_1, r_{i-1}, \ldots, q, r_{i+1}, \ldots, r_N | 0 < r_n < R_n, n = 1, 2, \ldots, i - 1, i + 1, \ldots, N\}$ and $B = \{h_\lambda(i, q) | \lambda = 1, \ldots, \mathrm{RW}(i)\}$. Then, it is obvious that the following equation will generate an one to one mapping from A to B using a fixed q:

$$l(r_1, r_2, \ldots, r_{i-1}, q, r_{i+1}, \ldots, r_N) = \sum_{n=1}^{i-1} r_n s_n + q s_i + \sum_{n=i+1}^{N} r_n s_n \quad (r_0 = 0 \ \text{ and } \ r_{N+1} = 0). \tag{17}$$

It is implicitly assumed that all the l's generated via (17) will be in ascending order by the mechanism described in the Appendix. It is also obvious to see that the cardinality of A equals the cardinality of B

$$\mathrm{Dim}(A) = \left(\prod_{n=1}^{i-1} R_i\right) \times 1 \times \left(\prod_{n=i+1}^{N} R_i\right) = \frac{1}{R_i} \prod_{n=1}^{N} R_i = \frac{L}{R_i} = \mathrm{RW}(i) = \mathrm{Dim}(B).$$

In the error gradient decent process, error will propagate along interconnection $V_{ih_\lambda(i,q)}(\lambda = 1, \ldots, \mathrm{RW}(i))$ from $\mu_{h_\lambda(i,q)}(\lambda = 1, \ldots, \mathrm{RW}(i))$ to x_i. The signal flow for the center of the first MF of x_1, namely $\partial E / \partial m_{10}$, is drawn in Fig. 6 (from Fig. 4) to demonstrate the approach of finding

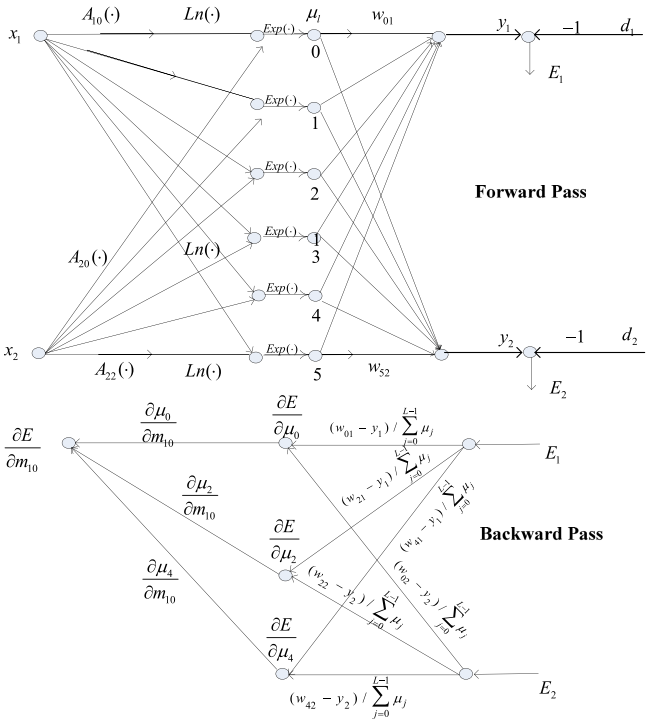


Fig. 6. Signal flow graph showing the gradient of error signal for the first MF of $x1$.

dependent and repeated links between input layer and hidden layer in F-CONFIS. For instance, in order to find $\partial E/\partial m_{10}$ of premise part in Fig. 5, we need to know that $V_{10} = V_{12} = V_{14} = \ln(A_{10}(x_1))$ are the links with the repeated weight of $\ln(A_{10}(x_1))$ from the first MF of x_1 to hidden neurons 0, 2, and 4.

Although (17) can indeed generate $\{h_\lambda(i,q)|\lambda = 1,\ldots,\mathrm{RW}(i)\}$ by considering all the combinations of r_i, it still not precise enough for the gradient descent training of F-CONFIS. The following Theorem 1 will show a precise formula of finding the ordinal indices of these links with repeated weights for the qth MF of the ith fuzzy input variable x_i.

Theorem 1: The ordinal indices $\{h_\lambda(i,q)|\lambda = 1,\ldots,\mathrm{RW}(i)\}$ for hidden neurons with links of repeated weights, which connect the qth MF of the ith fuzzy input variable x_i, can be found from the following equation:

$$h_\lambda(i,q) = (\lambda-1)\%s_i + qs_i + [(\lambda-1)/s_i]\,R_i s_i$$
$$(i = 1,\ldots,N;\ q = 0,\ldots,R_i-1;\ \lambda = 1,\ldots,\mathrm{RW}(i)) \quad (18)$$

where $A\ \%\ B$ represents the remainder of the division of A over B, and A/B represents the quotient of the integer division of A over B.

Proof: Please refer to the Appendix.

For illustration, the F-CONFIS in Fig. 4, $\mathrm{RW}(1) = 3$ implies that there are three repeated links with the same weight, which connect three hidden neurons and x_1. Therefore, from Theorem 1, the ordinal indices $\{h_\lambda(i,q)\,|i = 1; q = 0; \lambda = 1,2,3\}$ for hidden neurons with links of three repeated weights, which connect the 0th MF of the 1st fuzzy input variable x_1, is given as $(s_1 = 1)$

$$h_1(1,0) = (1-1)\%1 + 0*1 + (0/1)*2*1 = 0$$
$$h_2(1,0) = (2-1)\%1 + 0*1 + (1/1)*2*1 = 2$$
$$h_3(1,0) = (3-1)\%1 + 0*1 + (2/1)*2*1 = 4.$$

The ordinal indices $\{h_\lambda(i,q)|i = 1; q = 1; \lambda = 1,2,3\}$ connect the first MF of the 1st fuzzy input variable can be found as $(s_1 = 1)$

$$h_1(1,1) = (1-1)\%1 + 1*1 + (0/1)*2*1 = 1$$
$$h_2(1,1) = (2-1)\%1 + 1*1 + (1/1)*2*1 = 3$$
$$h_3(1,1) = (3-1)\%1 + 1*1 + (2/1)*2*1 = 5.$$

From Fig. 4, links carry $\ln(A_{10}(x_1))$ are connected to μ_0, μ_2, and μ_4 of hidden nodes, and links carry $\ln(A_{11}(x_1))$ are connected to μ_1, μ_3, and μ_5 of hidden nodes.

Therefore, from the above illustration, it is important to find the updating law of the repeated links for F-CONFIS, especially in a large neural fuzzy inference network. The following Theorem 2 will show a precise updating law of F-CONFIS.

Theorem 2: Applying the gradient descent algorithm for F-CONFIS, the gradient components of the premise part should be updated $\mathrm{RW}(i)$ times, where $\mathrm{RW}(i)$ is the total number of repeated links. In the consequent part, the gradient component is updated only one time. The gradient components are updated as

$$\begin{cases} m_{iq}(t+1) = m_{iq}(t) - \alpha \sum_{\lambda=1}^{\mathrm{RW}(i)} \frac{\partial E}{\partial \mu_{h_\lambda(i,q)}} \frac{\partial \mu_{h_\lambda(i,q)}}{\partial A_{iq}} \frac{\partial A_{iq}}{\partial m_{iq}} \\ \sigma_{iq}(t+1) = \sigma_{iq}(t) - \alpha \sum_{\lambda=1}^{\mathrm{RW}(i)} \frac{\partial E}{\partial \mu_{h_\lambda(i,q)}} \frac{\partial \mu_{h_\lambda(i,q)}}{\partial A_{iq}} \frac{\partial A_{iq}}{\partial \sigma_{iq}} \\ w_{jk}(t+1) = w_{jk}(t) - \beta \frac{\partial E(W)}{\partial w_{jk}} \end{cases} \quad (19)$$

and the gradient components of Gaussian-type learning algorithm are updated as

$$\begin{cases} m_{iq}(t+1) = m_{iq}(t) - \\ \quad \alpha \frac{1}{P} \sum_{\lambda=1}^{\mathrm{RW}(i)} \sum_{p=1}^{P} \sum_{k=1}^{M} \frac{\left(y_k^p - d_k^p\right)\left(w_{jk} - y_k^p\right)}{\sum_{j=0}^{L-1} \mu_j} \frac{\mu_{h_\lambda(i,q)}\left(x_i^p - m_{iq}\right)}{\sigma_{iq}^2} \\ \sigma_{iq}(t+1) = \sigma_{iq}(t) - \\ \quad \alpha \frac{1}{P} \sum_{\lambda=1}^{\mathrm{RW}(i)} \sum_{p=1}^{P} \sum_{k=1}^{M} \frac{\left(y_k^p - d_k^p\right)\left(w_{jk} - y_k^p\right)}{\sum_{j=0}^{L-1} \mu_j} \frac{\mu_{h_\lambda(i,q)}\left(x_i^p - m_{iq}\right)^2}{\sigma_{iq}^3} \\ w_{jk}(t+1) = w_{jk}(t) - \beta \frac{1}{P} \sum_{p=1}^{P} \left(y_k^p - d_k^p\right) \mu_j \Big/ \sum_{l=0}^{L-1} \mu_l \end{cases} \quad (20)$$

$$i = 1,\ldots,N,\ j = 0,\ldots,L-1,\ k = 1,\ldots,M,$$
$$q = 0,\ldots,R_i-1,\ \lambda = 1,\ldots,\mathrm{RW}(i).$$

Proof: By Proposition 1, we know that each variable x_i has $\mathrm{RW}(i)$ repeated links between the input layer and the hidden layer. Let $A_{iq}(q \in [1, R_i - 1])$ be the qth MF of the variable x_i, $\{\ell_\lambda = h_\lambda(i,q);\ \lambda = 1,\ldots,\mathrm{RW}(i)\}$ in (14) are the links with repeated weight A_{iq} for the qth MF of

x_i, and the weights of the repeated links are $V_{ih_\lambda(i,q)}(\lambda = 1, \ldots, \mathrm{RW}(i))$.

Let m_{iq} and σ_{iq} be the control parameters of A_{iq}. In the error gradient decent process, error will propagate along interconnection $V_{ih_\lambda(i,q)}(\lambda = 1, \ldots, \mathrm{RW}(i))$ to x_i. To get error's partial derivative with respect to m_{iq} and σ_{iq}, from (1), (2), (4), and (8) and by the chain rule, we can get these partial derivate as

$$\begin{cases} \dfrac{\partial E}{\partial \mu_{h_\lambda(i,q)}} \dfrac{\partial \mu_{h_\lambda(i,q)}}{\partial A_{iq}} \dfrac{\partial A_{iq}}{\partial m_{iq}} \\ \dfrac{\partial E}{\partial \mu_{h_\lambda(i,q)}} \dfrac{\partial \mu_{h_\lambda(i,q)}}{\partial A_{iq}} \dfrac{\partial A_{iq}}{\partial \sigma_{iq}}, \end{cases} \quad \lambda = 1, \ldots, \mathrm{RW}(i).$$

From (8), (9), (12), and (13), we have

$$\begin{cases} \dfrac{\partial E}{\partial m_{iq}} = \dfrac{1}{P} \displaystyle\sum_{p=1}^{P} \sum_{k=1}^{M} \dfrac{\left(y_k^p - d_k^p\right)\left(w_{jk} - y_k^p\right)}{\sum_{j=0}^{L-1} \mu_j} \dfrac{\mu_j \left(x_i^p - m_{iq}\right)}{\sigma_{iq}^2} \\ \dfrac{\partial E}{\partial \sigma_{iq}} = \dfrac{1}{P} \displaystyle\sum_{p=1}^{P} \sum_{k=1}^{M} \dfrac{\left(y_k^p - d_k^p\right)\left(w_{jk} - y_k^p\right)}{\sum_{j=0}^{L-1} \mu_j} \dfrac{\mu_j \left(x_i^p - m_{iq}\right)^2}{\sigma_{iq}^3} \end{cases}$$

$$i = 1, \ldots N; q = 0, \ldots, R_i - 1; j = 0, \ldots, L-1. \tag{21}$$

From (7) and (8), by chain rule, we have

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial y_k^p} \frac{\partial y_k^p}{\partial w_{jk}} = \frac{1}{P} \sum_{p=1}^{P} (y_k^p - d_k^p) \mu_j \Big/ \sum_{l=0}^{L-1} \mu_l$$
$$j = 0, \ldots, L-1; k = 1, \ldots, M. \tag{22}$$

After finding $\mathrm{RW}(i)$ by Proposition 1, all these intermediate partial derivatives should be summed up to give the final update for m_{iq} and σ_{iq} as the following equations:

$$\begin{cases} \dfrac{\partial E}{\partial m_{iq}} = \displaystyle\sum_{\lambda=1}^{\mathrm{RW}(i)} \dfrac{\partial E}{\partial \mu_{h_\lambda(i,q)}} \dfrac{\partial \mu_{h_\lambda(i,q)}}{\partial A_{iq}} \dfrac{\partial A_{iq}}{\partial m_{iq}} \\ \dfrac{\partial E}{\partial \sigma_{iq}} = \displaystyle\sum_{\lambda\lambda=1}^{\mathrm{RW}(i)} \dfrac{\partial E}{\partial \mu_{h_\lambda(i,q)}} \dfrac{\partial \mu_{h_\lambda(i,q)}}{\partial A_{iq}} \dfrac{\partial A_{iq}}{\partial \sigma_{iq}}. \end{cases} \tag{23}$$

In practical applications, given a hidden node number l, we can reversely find ordinal index of MFs for all fuzzy variable, namely $r_i(l)(i = 1, 2, \ldots, N)$ by (13). By (23), we can calculate one item in the right-hand side of (23) for every MF connected to hidden node. Traversing all the hidden nodes, we can find all the items in the right-hand side of (23) for every MFs. In consequent part, because there is no repeated links, the gradient component should be updated only one time. Therefore, the components of W are finally updated by the following equations:

$$\begin{cases} m_{iq}(t+1) = m_{iq}(t) - \alpha \displaystyle\sum_{\lambda=1}^{\mathrm{RW}(i)} \dfrac{\partial E}{\partial \mu_{h_\lambda(i,q)}} \dfrac{\partial \mu_{h_\lambda(i,q)}}{\partial A_{iq}} \dfrac{\partial A_{iq}}{\partial m_{iq}} \\ \sigma_{iq}(t+1) = \sigma_{iq}(t) - \alpha \displaystyle\sum_{\lambda=1}^{\mathrm{RW}(i)} \dfrac{\partial E}{\partial \mu_{h_\lambda(i,q)}} \dfrac{\partial \mu_{h_\lambda(i,q)}}{\partial A_{iq}} \dfrac{\partial A_{iq}}{\partial \sigma_{iq}} \\ w_{jk}(t+1) = w_{jk}(t) - \beta \dfrac{\partial E(W)}{\partial w_{jk}}. \end{cases}$$

From (19), (21), and (22), we have

$$\begin{cases} m_{iq}(t+1) = m_{iq}(t) - \\ \alpha \dfrac{1}{P} \displaystyle\sum_{\lambda=1}^{\mathrm{RW}(i)} \sum_{p=1}^{P} \sum_{k=1}^{M} \dfrac{\left(y_k^p - d_k^p\right)\left(w_{jk} - y_k^p\right)}{\sum_{j=0}^{L-1} \mu_j} \dfrac{\mu_{h_\lambda(i,q)} \left(x_i^p - m_{iq}\right)}{\sigma_{iq}^2} \\ \sigma_{iq}(t+1) = \sigma_{iq}(t) - \\ \alpha \dfrac{1}{P} \displaystyle\sum_{\lambda=1}^{\mathrm{RW}(i)} \sum_{p=1}^{P} \sum_{k=1}^{M} \dfrac{\left(y_k^p - d_k^p\right)\left(w_{jk} - y_k^p\right)}{\sum_{j=0}^{L-1} \mu_j} \dfrac{\mu_{h_\lambda(i,q)} \left(x_i^p - m_{iq}\right)^2}{\sigma_{iq}^3} \\ w_{jk}(t+1) = w_{jk}(t) - \beta \dfrac{1}{P} \displaystyle\sum_{p=1}^{P} \left(y_k^p - d_k^p\right) \mu_j \Big/ \sum_{l=0}^{L-1} \mu_l. \end{cases}$$

Q.E.D.

For example, the number of repeated links $\mathrm{RW}(1)$ is 3 by Proposition 1. According to Theorem 2, the gradient components m_{10} of the premise part should be updated $\mathrm{RW}(1)$ times. Therefore, we have

$$\begin{aligned} m_{10}(t+1) &= m_{10}(t) - \alpha \sum_{s=1}^{3} \frac{\partial E}{\partial \mu_{h_\lambda(1,0)}} \frac{\partial \mu_{h_\lambda(1,0)}}{\partial A_{10}} \frac{\partial A_{10}}{\partial m_{10}} \\ &= m_{10}(t) - \alpha \left(\frac{\partial E}{\partial \mu_0} \frac{\partial \mu_0}{\partial A_{10}} \frac{\partial A_{10}}{\partial m_{10}} + \frac{\partial E}{\partial \mu_2} \frac{\partial \mu_2}{\partial A_{10}} \frac{\partial A_{10}}{\partial m_{10}} \right. \\ &\quad \left. + \frac{\partial E}{\partial \mu_4} \frac{\partial \mu_4}{\partial A_{10}} \frac{\partial A_{10}}{\partial m_{10}} \right). \end{aligned}$$

For notational convenience, let $m_{ij}(\alpha) = m_{ij}(t+1)$, $\sigma_{ij}(\alpha) = \sigma_{ij}(t+1), W(\alpha, \beta) = W(t+1), [\Delta m_{ij}, \Delta\sigma_{ij}, \Delta w_{jp}] = [\partial E/\partial m_{ij}, \partial E/\partial \sigma_{ij}, \partial E/\partial w_{jk}]$. We can explicitly express weights and error function in terms of learning rates. From (12), we have

$$W(\alpha, \beta) = W - [\alpha \Delta m_{ij}, \alpha \Delta \sigma_{ij}, \beta \Delta w_{jk}]. \tag{24}$$

After weights are updated, we can get new error with respect to new weights $E(\alpha, \beta) = E(W(\alpha, \beta))$. Gradient descent algorithm updates weights in negative gradient direction $-g = -[\Delta m_{ij}, \Delta \sigma_{ij}, \Delta w_{jk}]$. The weights update is $\Delta W(\alpha, \beta) = -[\alpha \Delta m_{ij}, \alpha \Delta \sigma_{ij}, \beta \Delta w_{jk}]$. To improve convergence rate, it is important to find the optimal learning rate. The following Theorem 3 will show the optimal learning rate of neuro-fuzzy systems.

Theorem 3: The optimal learning rate β_{opt} of the consequent part in neuro-fuzzy systems can be obtained as

$$\beta_{\text{opt}} = \frac{\sum_{p=1}^{P}\sum_{k=1}^{K}\left(y_k^p(\alpha) - d_k^p\right)\Delta y_k(\alpha)}{\sum_{p=1}^{P}\sum_{k=1}^{K}\Delta y_k(\alpha)\Delta y_k(\alpha)} \tag{25}$$

where $\Delta y_k(\alpha) = \sum_{j=1}^{L}\mu_j(\alpha)\Delta w_{jp}/s(\alpha)$, and the near optimal learning rate α_{opt} of the premise part in neuro-fuzzy systems can be obtained

$$\alpha_{\text{opt}} = -\frac{\frac{\partial E(\alpha)}{\partial \alpha}\,|_{\alpha=0}}{\frac{\partial^2 E(\alpha)}{\partial \alpha^2}\,|_{\alpha=0}}. \tag{26}$$

Proof: From (8), we have

$$E(W) = \frac{1}{2P}\sum_{p=1}^{P}\sum_{k=1}^{M}\left(y_k^p - d_k^p\right)^2 \geq 0.$$

In F-CONFIS, given the MFs are continuously differentiable and provided that learning rates is greater than zero, the error of the gradient-descent learning process will be decreased. Taking error gradient with respect to (α, β) is zero, and then the near optimal learning rates satisfy

$$\frac{\partial E(\alpha,\beta)}{\partial \alpha}\,|_{\alpha=\alpha_{\text{opt}}} = 0 \quad \frac{\partial E(\alpha,\beta)}{\partial \beta}\,|_{\beta=\beta_{\text{opt}}} = 0. \tag{27}$$

Form (7), (8), and (22), we get

$$\Delta w_{jk} = \frac{1}{PM}\sum_{p=1}^{P}\sum_{k=1}^{M}\frac{\left(y_k^p - d_k^p\right)\mu_j}{s}. \tag{28}$$

Denote $\Delta y_k(\alpha) = \sum_{j=1}^{L}\mu_j(\alpha)\Delta w_{jk}/s(\alpha)$, from (7), we have

$$\begin{aligned} y_k(\alpha,\beta) &= \frac{\sum_{j=1}^{L}\mu_j(\alpha)w_{jk}(\beta)}{s(\alpha)} \\ &= \frac{\sum_{j=1}^{L}\mu_j(\alpha)\left(w_{jk} - \beta\Delta w_{jk}\right)}{s(\alpha)} \\ &= y_k(\alpha) - \beta\Delta y_k(\alpha). \end{aligned} \tag{29}$$

Differentiating (8) with respect to β, we get

$$\frac{\partial E(\alpha,\beta)}{\partial \beta} = \frac{1}{P}\sum_{p=1}^{P}\sum_{k=1}^{K}\left(y_k^p(\alpha,\beta) - d_k^p\right)\frac{\partial y_k^p(\alpha,\beta)}{\partial \beta}. \tag{30}$$

From (29) and (30), we have

$$\frac{\partial E(\alpha,\beta)}{\partial \beta} = -\frac{1}{P}\sum_{p=1}^{P}\sum_{k=1}^{K}\left(y_k^p(\alpha) - \beta\Delta y_p(\alpha) - d_k^p\right)\Delta y_k(\alpha). \tag{31}$$

From (27) and (31), we get the optimal learning rate β_{opt} of the consequent part

$$\beta_{\text{opt}} = \frac{\sum_{p=1}^{P}\sum_{k=1}^{K}\left(y_k^p(\alpha) - d_k^p\right)\Delta y_k(\alpha)}{\sum_{p=1}^{P}\sum_{k=1}^{K}\Delta y_k(\alpha)\Delta y_k(\alpha)}.$$

Although the optimal learning rate β_{opt} of the consequent part can be obtained, unfortunately, the analytical expression of the premise part cannot be solved in a similar way because the premise part, $\partial E(\alpha,\beta)/\partial\alpha = 0$, is usually a transcendental equation. To obtain α_{opt}, line search method [31] is feasible. The simplest way for performing a line search is to take a series of small steps along the chosen search direction until the error increases, and then go back one step. Though there are other better approaches for performing inexact line searches such as Armijo line search [32], [33], it is still time consuming. In this paper, we propose a method to get the near optimal learning rate for α by expanding $E(\alpha,\beta)$ with Taylor series. To avoid the complicacy of multivariable Taylor series, we let the weights of consequent part w_{jk} stay the same (that is to let $\beta = 0$) while deriving the learning rate of the premise part. Taylor series of $E(\alpha) = E(\alpha, 0)$ to second order around point 0 is

$$E(\alpha) = E(0) + \alpha\frac{\partial E(\alpha)}{\partial \alpha}\bigg|_{\alpha=0} + \frac{\alpha^2}{2}\frac{\partial^2 E(\alpha)}{\partial \alpha^2}\bigg|_{\alpha=0} + O(\alpha^3). \tag{32}$$

If we neglect the higher order terms and differentiate both side of (32) with respect to α, we have

$$\frac{\partial E(\alpha)}{\partial \alpha} = \frac{\partial E(\alpha)}{\partial \alpha}\bigg|_{\alpha=0} + \alpha\frac{\partial^2 E(\alpha)}{\partial \alpha^2}\bigg|_{\alpha=0}. \tag{33}$$

From (27) and (33), we get the near optimal learning rate α_{opt} of the consequent part

$$\alpha_{\text{opt}} = -\frac{\frac{\partial E(\alpha)}{\partial \alpha}\Big|_{\alpha=0}}{\frac{\partial^2 E(\alpha)}{\partial \alpha^2}\Big|_{\alpha=0}}.$$

Q.E.D.

Next, we need to derive $\partial E(\alpha)/\partial\alpha$ and $\partial^2 E(\alpha)/\partial\alpha^2$. Let

$$e_k^p = \frac{1}{2}\left(y_k^p - d_k^p\right)^2. \tag{34}$$

Then

$$E = \frac{1}{P}\sum_{p=1}^{P}\sum_{k=1}^{M}e_k^p \tag{35}$$

$$\begin{cases} \dfrac{\partial E(\alpha)}{\partial \alpha} = \dfrac{1}{P}\displaystyle\sum_{p=1}^{P}\sum_{k=1}^{M}\frac{\partial e_k^p(\alpha)}{\partial \alpha} \\ \dfrac{\partial^2 E(\alpha)}{\partial \alpha^2} = \dfrac{1}{P}\displaystyle\sum_{p=1}^{P}\sum_{k=1}^{M}\frac{\partial^2 e_k^p(\alpha)}{\partial \alpha^2}. \end{cases} \tag{36}$$

For conciseness, we use $\partial e_k(\alpha)/\partial\alpha$ and $\partial^2 e_k(\alpha)/\partial\alpha^2$ to express $\partial e_k^p(\alpha)/\partial\alpha$ and $\partial^2 e_k^p(\alpha)/\partial\alpha^2$ corresponding to any pth training pattern.

From (34), we get

$$\frac{\partial e_k(\alpha)}{\partial\alpha} = (y_k(\alpha) - d_k)\frac{\partial y_k(\alpha)}{\partial\alpha}. \tag{37}$$

Differentiating (37) with respect to α, we get

$$\frac{\partial e_k^2(\alpha)}{\partial\alpha^2} = \left(\frac{\partial y_k(\alpha)}{\partial\alpha}\right)^2 + (y_k(\alpha) - d_k)\frac{\partial^2 y_k(\alpha)}{\partial\alpha^2}. \tag{38}$$

Differentiating (7) with respect to α, we get

$$\frac{\partial y_k(\alpha)}{\partial\alpha} = \frac{\sum \frac{\partial(\mu_j(\alpha)w_{jk})}{\partial\alpha} - \left(\sum \frac{\partial(\mu_j(\alpha))}{\partial\alpha}\right)y_k(\alpha)}{s(\alpha)}. \tag{39}$$

Differentiating (39) with respect to α, we have

$$\frac{\partial^2 y_k(\alpha)}{\partial\alpha^2} = \frac{\sum w_{jk}\frac{\partial^2\mu_j(\alpha)}{\partial\alpha^2} - y_k(\alpha)\sum\frac{\partial^2(\mu_j(\alpha))}{\partial\alpha^2} - 2\frac{\partial y_k(\alpha)}{\partial\alpha}\sum\frac{\partial(\mu_j(\alpha))}{\partial\alpha}}{s(\alpha)}. \tag{40}$$

Form (1) and (9), we have

$$\mu_j(\alpha) = \prod_{i=1}^{n} A_{ij}(\alpha) = \exp\left(-\sum_{i=1}^{n}\left(\frac{(x_i - m_{ij} + \alpha\Delta m_{ij})^2}{2(\sigma_{ij} - \alpha\sigma_{ij})^2}\right)\right). \tag{41}$$

Differentiating (41) with respect to α, we have

$$\frac{\partial\mu_j(\alpha)}{\partial\alpha} = -\mu_j(\alpha)\sum_{i=1}^{n}\left(\frac{(x_i - m_{ij}(\alpha))\Delta m_{ij}}{(\sigma_{ij}(\alpha))^2} + \frac{(x_i - m_{ij}(\alpha))^2\Delta\sigma_{ij}}{(\sigma_{ij}(\alpha))^3}\right). \tag{42}$$

Differentiating (42) with respect to α, we have

$$\frac{\partial^2\mu_j(\alpha)}{\partial\alpha^2} = -\frac{\partial\mu_j(\alpha)}{\partial\alpha} \times\sum_{i=1}^{n}\left(\frac{(x_i - m_{ij}(\alpha))(\sigma_{ij}(\alpha)\Delta m_{ij} + (x_i - m_{ij}(\alpha))\Delta\sigma_{ij})}{(\sigma_{ij}(\alpha))^3} - \mu_j(\alpha)h(\alpha)\right) \tag{43}$$

where $h(\alpha)$ is

$$h(\alpha) = \frac{\partial}{\partial\alpha}\sum_{i=1}^{n}\left(\frac{(x_i - m_{ij}(\alpha))\Delta m_{ij}}{(\sigma_{ij}(\alpha))^2} + \frac{(x_i - m_{ij}(\alpha))^2\Delta\sigma_{ij}}{(\sigma_{ij}(\alpha))^3}\right) = \sum_{i=1}^{n}\left(\frac{(\Delta m_{ij})^2}{(\sigma_{ij}(\alpha))^2} + \frac{4(x_i - m_{ij}(\alpha))\Delta m_{ij}\Delta\sigma_{ij}}{(\sigma_{ij}(\alpha))^3} + \frac{3(x_i - m_{ij}(\alpha))^2(\Delta\sigma_{ij})^2}{(\sigma_{ij}(\alpha))^4}\right).$$

By (42) and (43), we can calculate $\partial\mu_j(\alpha)/\partial\alpha$ and $\partial^2\mu_j(\alpha)/\partial\alpha^2$. With $\partial\mu_j(\alpha)/\partial\alpha$ and $\partial^2\mu_j(\alpha)/\partial\alpha^2$, by (39) and (40), we can calculate $\partial y_k(\alpha)/\partial\alpha$ and $\partial^2 y_k(\alpha)/\partial\alpha^2$. With $\partial y_k(\alpha)/\partial\alpha$ and $\partial^2 y_k(\alpha)/\partial\alpha^2$, by (36)–(38), we can calculate $\partial E(\alpha)/\partial\alpha$ and $\partial^2 E(\alpha)/\partial\alpha^2$. With $\partial E(\alpha)/\partial\alpha$ and $\partial^2 E(\alpha)/\partial\alpha^2$, by (26), we can calculate α_{opt}. For each iteration, during the gradient descent process, the near optimal learning rate of the premise part α_{opt} is first calculated via (26). With the α_{opt}, the control parameters of MFs are updated. Then, the optimal learning rate of the consequent part is calculated by (25). Therefore, according to Theorem 2, the components of W are finally updated by following equations:

$$\begin{cases} m_{iq}(t+1) = m_{iq}(t) - \alpha_{\text{opt}} \times\left(\sum_{\lambda=1}^{\text{RW}(i)}\frac{\partial E}{\partial\mu_{h_\lambda(i,q)}}\frac{\partial\mu_{h_\lambda(i,q)}}{\partial A_{iq}}\frac{\partial A_{iq}}{\partial m_{iq}}\right) \\ \sigma_{iq}(t+1) = \sigma_{iq}(t) - \alpha_{\text{opt}} \times\left(\sum_{\lambda=1}^{\text{RW}(i)}\frac{\partial E}{\partial\mu_{h_\lambda(i,q)}}\frac{\partial\mu_{h_\lambda(i,q)}}{\partial A_{iq}}\frac{\partial A_{iq}}{\partial\sigma_{iq}}\right) \\ w_{jk}(t+1) = w_{jk}(t) - \beta_{\text{opt}}\ \frac{\partial E(W)}{\partial w_{jk}}. \end{cases} \tag{44}$$

From (44), we have, (45), as shown at the bottom of the next page.

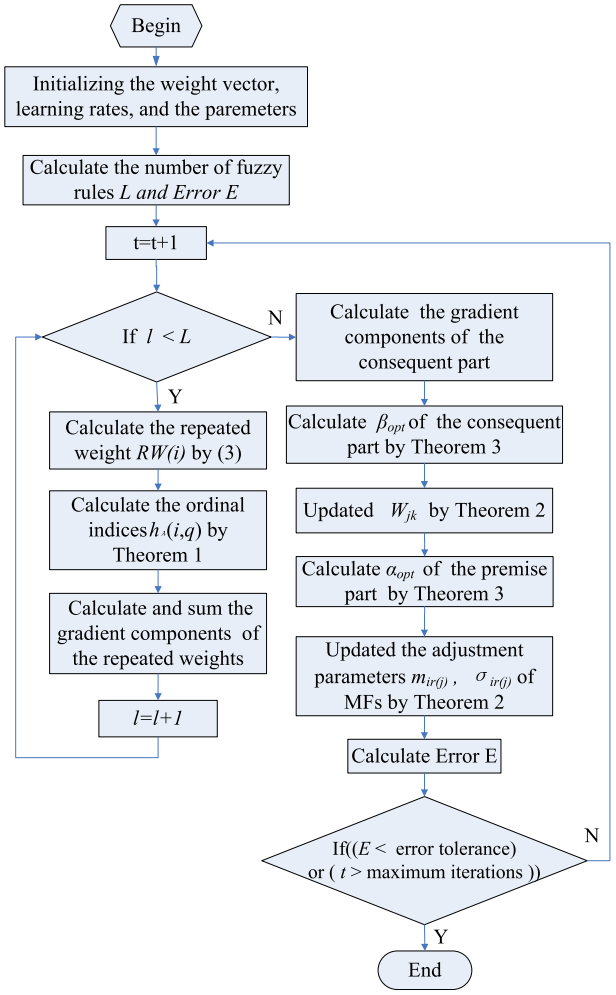


Fig. 7. Training process of the optimal learning of neuro-fuzzy system.

Algorithm 1 Optimal Learning of Neuro-Fuzzy System via F-CONFIS

Step 1: Initializing the weight vector, learning rates. Let t be iteration count, $t = 1$.

Step 2: Calculate the number of fuzzy rule L by (6), truth value μ_l by (1), output y_k by (7), error E in (8), the repeated link weights RW(i) by(3), and the ordinal indices $h_\lambda(i, q)$ by Theorem 1.

Step 3: Calculates the update changes of the control parameters in the premise part i.e., the overall update changes of the centers and spreads of the Gaussian MFs.

1) Let hidden node number $l = 0$.
2) For training input data $i = 1, \ldots, N$
   a) By (13), get the ordinal index $r_i(l)$ of MFs for the ith fuzzy variable.
   b) Find the gradient components $\partial E/\partial m_{ir_i(l)}$ and $\partial E/\partial \sigma_{ir_i(l)}$ in premise part via (21).
   c) Since every same weight A_{iq} will be repeated for RW(i) times, we have to sum the update changes with the same subscripts for ith fuzzy variable x_i for RW(i) times by (23).
3) If l less than to the total number of fuzzy rules L, then $l = l + 1$, go to 2), else go to next step 4.

Step 4: Calculate the gradient components of consequent part by (22).

Step 5: Calculate the learning rate α_{opt} of the premise part via (26) according to Theorem 3.

Step 6: Update m_{ij} and σ_{ij} by (45) according to Theorem 2.

Step 7: Calculate the learning rate β_{opt} of the consequent part by (25) according to Theorem 3.

Step 8: Update w_{jk} by (45) according to Theorem 2.

Step 9: Calculate error E in (8).

Step 10: If error E is less than error tolerance or iteration t reaches maximum iterations, go to step 11, else iteration $t = t + 1$; go to step 2.

Step 11: Stop.

The optimal learning rates of the premise part and the consequent part are given by Theorem 3. Next, an optimal learning algorithm for F-CONFIS will be described.

Fig. 7 describes the process of the proposed learning.

IV. Discussion of Time Complexity and Derivative-Based Method

Assume there are n adjustable parameters, L fuzzy rules. The training procedure of FIS can be summarized as the following pseudo-code.

```
while(epochs< max_epochs)
{
For(i=0; i< L; i++)//forward pass of training algorithm
{
Calculate_firing_strength_of_each_rule( );
Calculate_the_output_of_FIS( );
}
if(error< tolerance)break;
//backward pass of training algorithm
Calculate_derivative ( );
Update_the_parameters( );
}
```

In one epoch, the time complexity for forward pass is $O(L)$. There are three cases for backward pass.

1) If only first-order derivative is used, then the time complexity is $O(n)$.
2) If second-order derivative is used, then the time complexity is $O(n^2)$.
3) If there are operation of inverse matrix [like Levenberg–Marquardt (LM) method or Gauss–Newton (QN) method], then the time complexity is $O(n^3)$.

So, the overall time complexity for algorithm may be O[epochs*$(L+n)$], O(epochs*$(L + n^2)$), or O(epochs*$(L+n^3)$). ANFIS and our proposed method used only the first-order derivative, the time complexity is O(epochs*$(L + n)$). For a real application, the n and L are fixed, whereas epochs is variable. It usually holds that $L < n <$ epochs, so the dominated factor is epochs. The proposed approach applies dynamic optimal learning algorithm in premise and consequent parts of FIS, so it can obtain less iteration times than that of ANFIS, which leading to reduce of epochs and time complexity. Zhao–Li–Irwin's method apply LM method [34], LM method contains matrix inverse operation, so its time complexity is O(epochs*$(L+n^3)$). Theoretically, LM method can achieve less iteration times than gradient-descent method, but n^3 item in time complexity counteract this advantage.

The derivative-based optimization can be written as

$$W_{k+1} = W_k - \eta_k B_k \nabla E_k. \tag{46}$$

$$\begin{cases} m_{iq}(t+1) = m_{iq}(t) - \alpha_{\text{opt}} \left(\dfrac{1}{P} \displaystyle\sum_{\lambda=1}^{\text{RW}(i)} \sum_{p=1}^{P} \sum_{k=1}^{M} \dfrac{\left(y_k^p - d_k^p\right)\left(w_{jk} - y_k^p\right)}{\sum_{j=0}^{L-1} \mu_j} \dfrac{\mu_{h_\lambda(i,q)}\left(x_i^p - m_{iq}\right)}{\sigma_{iq}^2} \right) \\ \sigma_{iq}(t+1) = \sigma_{iq}(t) - \alpha_{\text{opt}} \left(\dfrac{1}{P} \displaystyle\sum_{\lambda=1}^{\text{RW}(i)} \sum_{p=1}^{P} \sum_{k=1}^{M} \dfrac{\left(y_k^p - d_k^p\right)\left(w_{jk} - y_k^p\right)}{\sum_{j=0}^{L-1} \mu_j} \dfrac{\mu_{h_\lambda(i,q)}\left(x_i^p - m_{iq}\right)^2}{\sigma_{iq}^3} \right) \\ w_{jk}(t+1) = w_{jk}(t) - \beta_{\text{opt}} \left(\dfrac{1}{P} \displaystyle\sum_{p=1}^{P} \left(y_k^p - d_k^p\right)\mu_j \Big/ \sum_{l=0}^{L-1} \mu_l \right) \end{cases}$$

$$i = 1, \ldots, N,\ j = 0, \ldots, L-1,\ k = 1, \ldots, M,\ q = 0, \ldots, R_i - 1,\ \lambda = 1, \ldots, \text{RW}(i). \tag{45}$$

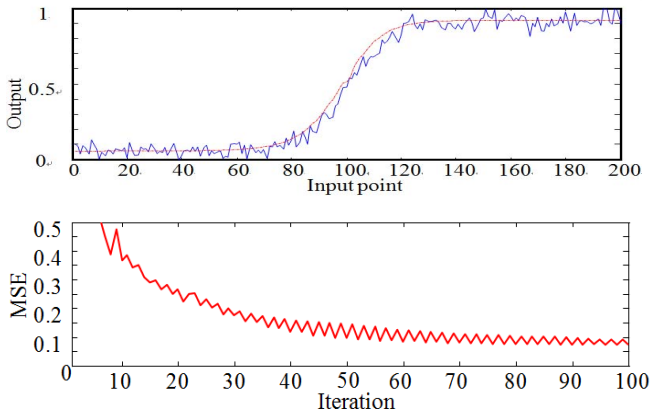


Fig. 8. Conventional method of genfis1.

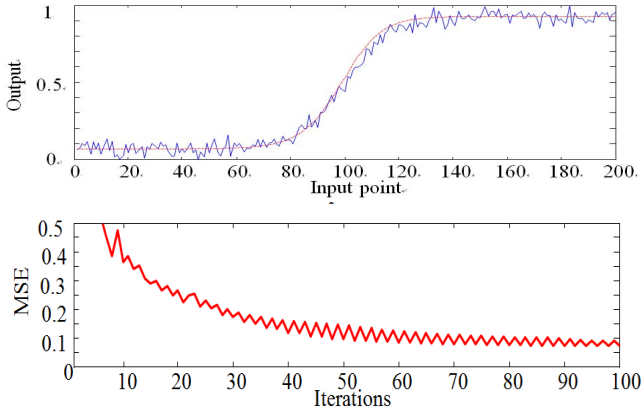


Fig. 9. Conventional method of genfis2.

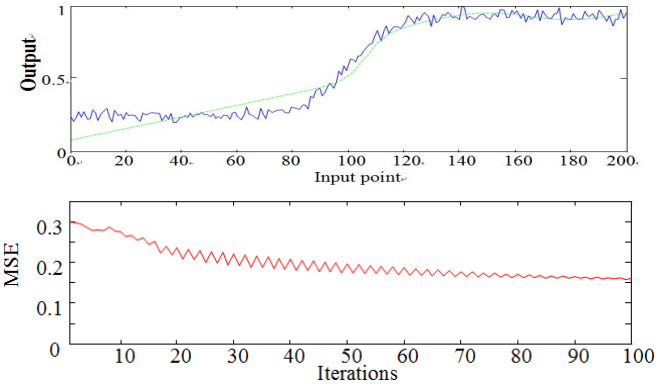


Fig. 10. Conventional method of genfis3.

When B_k is identity matrix ($B_k = I_k$), it is gradient descent method. When B_k is the inverse of Hessian matrix ($B_k = (\nabla^2(w_k))^{-1}$), it is Newton's method. Since the inverse of Hessian matrix may not exist and the computation cost is high. The quasi-Newton method is a practical alternative, in which B_k is an approximation of the true inverse Hessian matrix. BFGS and DFP are two kinds of quasi-Newton method. Theoretically, if the object function is quadratic, the convergence rate of quasi-Newton method is superlinear. If we take the BFGS or DFP method, then we cannot get the optimal learning rate in closed form, and must resort to line search method. If the line search is inexact or not based on the Wolfe conditions, the B_k will becomes a poor approximation to the true inverse Hessian and produce bad results. The possible exploration in dynamic learning rate of approximate second-order derivative methods like BFGS or DFP for a neuro-fuzzy system should be discouraged and can be regarded as a possible direction of future work.

V. Illustrated Examples

In practical application, choosing the proper learning rate is a time consuming process. Manually selected learning rate needs many experiments and the result may lack generality. The proposed optimal learning algorithm can adjust the learning rate dynamically. In this section, several examples are conducted to verify the effectiveness of the proposed training algorithms for the F-CONFIS, and compared with the results from some well-known methods. In our proposed approach, the values of premise parameters are initialized using fuzzy C-means clustering method, and the values of consequent parameters are initialized by least-squares method.

Example 1 *(Comparison of the Performance of Well-Known Fuzzy Systems With Conventional Method and Our Proposed Method):* Consider the approximation of the following 1-D nonlinear function [35]

$$y(x) = \frac{1}{1 + e^{(-25(x-0.5))}} \quad 0 \le x \le 1 \tag{47}$$

where x is input and y is output. In all, 400 data samples were generated, every sample is $(x(t), y(t))$. The first 200 data samples is used for the training process and remaining 200 for testing process. $x(t)$ uniformly distributed in the interval [0, 1] and $y(t)$ was given by

$$y(t) = f(x(t)) + \varepsilon(t) \quad 0 \le x \le 1 \tag{48}$$

where $\varepsilon(t) \sim N(0, 0.04^2)$.

Example 1.1 (Performance Comparison Between ANFIS and F-CONFIS): For comparison purpose, fuzzy models were developed under the same condition using several well-known methods, including genfis1, genfis2, and genfis3 of ANFIS and F-CONFIS. The fuzzy models were optimized by the proposed method and the conventional algorithms of ANFIS, which are available in MATLAB toolbox.

After iteration reaches 100, Figs. 8–10 show the outputs of genfis1, genfis2, and genfis3 trained by conventional method of ANFIS. Fig. 11 shows the output of F-CONFIS trained by the proposed method. In each figure, top curve is the output (dashed line) of neuro-fuzzy systems and training data (solid line) and bottom curve is the prediction error between the output and original training data. It is obvious that proposed method achieves better accuracy and faster convenience speed.

Table II shows the performance (MSE) comparison of conventional methods and proposed method. Both conventional and the new optimal learning approaches were utilized in the example, then 10, 100, 200, 500, and 1000 iterations were conducted for the methods. It can be seen that when iteration reaches 10, the MSE of proposed method is 0.010832, the

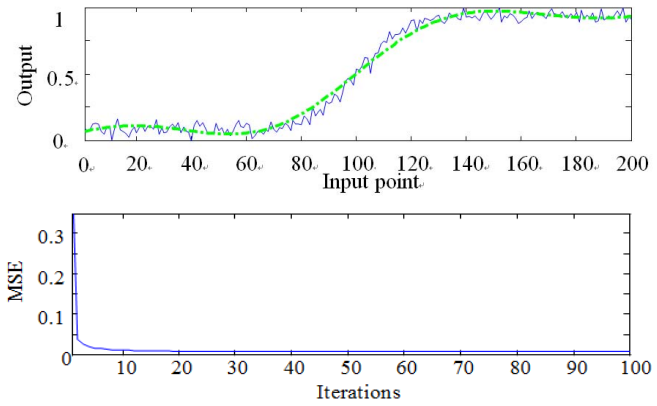


Fig. 11. Proposed method.

TABLE II
COMPARISON PERFORMANCE OF OTHER METHODS AND OUR METHOD

Iterations/ Method	ANFIS			F-CONFIS
	genfis1	genfis2	genfis3	Proposed method
10	0.348	0.3449	0.2154	0.010832
100	0.0768	0.8929	0.1146	0.008234
200	0.0705	0.194	0.1011	0.008061
500	0.0678	0.1727	0.0772	0.005762
1000	0.0637	0.0665	0.0791	0.0038265

TABLE III
COMPARISON PERFORMANCE OF FIXED LEARNING RATE AND OUR METHOD

Iterations/ Method	α= 0.2, β= 0.1	α= 0.5, β= 0.8	α= 0, β= 0.6	α= 0.4, β= 0.4	Proposed method
10	0.093	0.172	0.092	0.116	0.00098
100	0. 097	0.204	0.055	0.303	0.00097
200	0. 077	0.076	0.050	0.036	0.00093
500	0.039	0.124	0.057	0.311	0.00082
1000	0. 036	0.061	0.056	0.140	0.00079

TABLE IV
COMPARISON OF THE ACCURACY AND CONVERGENCE OF OTHER METHOD AND OUR METHOD

Method	Iterations	MSE
Conventional Approach	500	0.002306
Zhao-Li-Irwin (2013)	50	0.00129
The Proposed Method	10	0.00098454

error reduction is is much better than that of other methods at iteration of 1000.

Example 1.2 (Comparison of Conventional Algorithms and Our Proposed Algorithm): For comparison purpose, the experiments were conducted under the same condition. ANFIS and F-CONFIS are trained using the conventional gradient descent algorithm with fixed learning rate and the proposed method respectively.

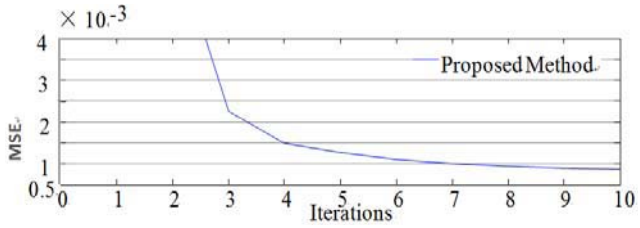


Fig. 12. MSE value and convergence of the proposed method.

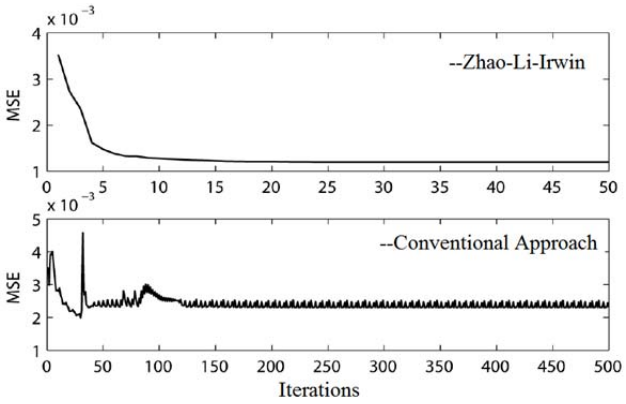


Fig. 13. Comparison of the accuracy and convergence of using the Zhao–Li–Irwin approach in 2013 (upper) and the conventional one (bottom).

Table III shows the performance comparison of the proposed method and traditional gradient descent with fixed learning rate algorithm (Case a: $\alpha = 0.2, \beta = 0.1$; Case b: $\alpha = 0.5$, $\beta = 0.8$; Case c: $\alpha = 0, \beta = 0.6$; Case d: $\alpha = 0.4$, $\beta = 0.4$). From Table III, we can see that when iteration reaches 10, the proposed method almost converge to zero. The error reduction is much better than that of others method at iteration of 1000. The improvement of convergence rate is considerable and impressive.

Example 1.3 (Comparison of Zhao–Li–Irwin' New Algorithm of ANFIS and Our Proposed Learning of F-CONFIS): For comparison purpose, the experiments were conducted under the same condition. Zhao–Li–Irwin' method [34] is a new gradient descent training algorithm for fuzzy neural models in 2013. In this example, we conduct the comparison among the conventional gradient descent method, Zhao–Li–Irwin' method of ANFIS, and our proposed approach of F-CONFIS.

After 10, 50, and 500 iterations, the results are shown in Table IV. It can be seen that the MSE is 0.00129 in Zhao's gradient descent approach when iteration reaches 50, whereas the MSE of the proposed method is 0.00098454 when iteration reaches 10, the convergence speed of proposed method is much faster. Figs. 12 and 13 show the curve of MSE value of the three methods. It is shown that the proposed approach achieves excellent performance.

Example 2 *(Nonlinear System Identification):* The second example is to identify a nonlinear system, which is described

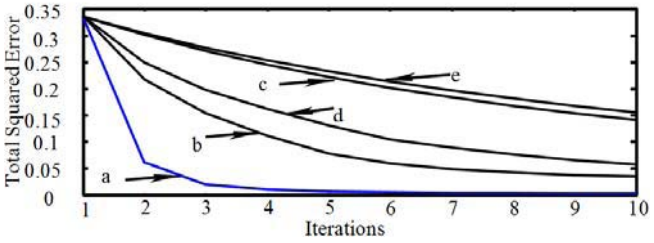


Fig. 14. Performance comparison for Example 1. Case a: our algorithm. Case b: $\alpha = 0.5$, $\beta = 0.8$. Case c: $\alpha = 0$, $\beta = 0.6$. Case d: $\alpha = 0.4$, $\beta = 0.4$. Case e: $\alpha = 0.1$, $\beta = 0.2$.

by the second-order difference equation [36]

$$y(k+1) = g[y(k), y(k-1)] + u(k) \tag{49}$$

where

$$g[y(k), y(k-1)] = \frac{y(k)y(k+1)[y(k)+2.5]}{1+y^2(k)+y^2(k-1)}. \tag{50}$$

A series-parallel neuro-fuzzy system identifier [18], [36] is given by

$$\hat{y}(k+1) = \hat{f}[y(k), y(k-1)] + u(k) \tag{51}$$

where $\hat{f}[y(k), y(k-1)]$ is the form of (7) with two fuzzy variables $y(k)$ and $y(k-1)$. The 500 training sample are generated by the plant model using a random input signal $u(k)$, which is distributed in $[-2, 2]$ and every fuzzy variable has five MFs. The MFs are the Gaussian functions. After the training process is finished, the model is tested by applying a sinusoidal input signal $u(k) = \sin(2\pi k/25)$. The proposed optimal learning algorithm was applied to the F-CONFIS.

Example 2.1 (Comparison of Fixed Learning Rates and Dynamic Learning Rate in F-CONFIS): For comparison purpose, the experiments are conducted in F-CONFIS with fixed learning rates and dynamic learning rate.

After 500 iterations, the trajectories of error of first 10 iterations are shown in Fig. 14. It also shows the performance comparison with the Cases b–e, where the learning rates are fixed (Case a: proposed method; Case b: $\alpha = 0.5, \beta = 0.8$; Case c: $\alpha = 0$, $\beta = 0.6$; Case d: $\alpha = 0.1$, $\beta = 0.2$). Apparently, the error of dynamic approach converges to zero faster than other cases.

Table V shows the comparison performance with fixed learning rate and the proposed method, where 10, 200, and 500 iterations (t) were conducted. It can be seen that the total squared error J of proposed method is 0.00074829 when iteration reaches 10, whereas the total square error J of other cases is close to this value till iteration reaches 200. Obviously, the proposed tuning algorithm for F-CONFIS is much better than other cases.

Example 2.2 (Comparison of Conventional Optimal Learning Methods of ANFIS and the Proposed Algorithm of F-CONFIS): To improve convergence rate, some kinds of dynamic learning rate methods [12], [16], [18]–[22] can be applied to the neuro-fuzzy systems. The learning rate of premise part was obtained by genetic search techniques in [18].

TABLE V
PERFORMNCE COMPARISON OF GRADIENT DESCENT ALGORITHM WITH FIXED LEARNING RATE AND PROPOSED ALGORITHM ($E^{-2} = \times 10^{-2}$)

Method	α_{opt}	β_{op}	J t=10	J t=200	J t=500
Proposed method	dynamic	dynamic	0. 07E^{-2}	0.03 E^{-2}	0. 025 E^{-2}
case b	0.5	0.8	3.62 E^{-2}	0.82 E^{-2}	0.241 E^{-2}
case c	0	0.6	14.2 E^{-2}	0.354E^{-2}	0.211 E^{-2}
case d	0.4	0.4	5.79 E^{-2}	1.363E^{-2}	0.863 E^{-2}
case e	0.1	0.2	15.6 E^{-2}	0.795E^{-2}	0.338 E^{-2}

TABLE VI
COMPARISON WITH CONVENTIONAL OPTIMAL LEARNING

Method	α_{opt}	β_{opt}	J $t=5$	J $t=120$
Proposed method	dynamic	dynamic	0.003286	0.0004019
Conventional Optimal Learning Method	Genetic algorithm	dynamic	0.00719	0.0023

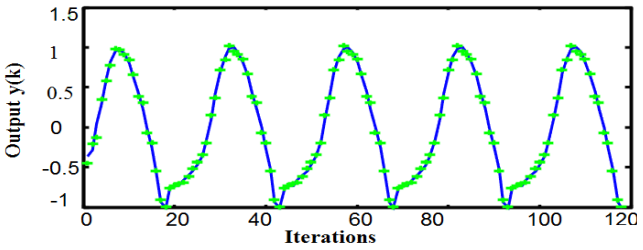


Fig. 15. Output of F-CONFIS trained by gradient decent with dynamic learning rates method, the solid line is the original data and the sign + denotes the F-CONFIS output.

This example conducts the performance comparison between the conventional dynamic optimal learning method in [18] and our proposed method under the same condition. Table VI shows the performance comparison between conventional optimal learning methods of the fuzzy NN in [18] with the proposed method.

After five iterations and 120 iterations, the total squared error J of the proposed method is 0.003286 and 0.0004019 compared with 0.00719 and 0.0023 of method in [18]. It can be seen that the convergence speed of the proposed algorithm is dozens times faster than those of others. To show the accuracy of the proposed method, the output of F-CONFIS trained by proposed method is shown in Fig. 15, where the sign + denotes the F-CONFIS output and the solid line is the original data.

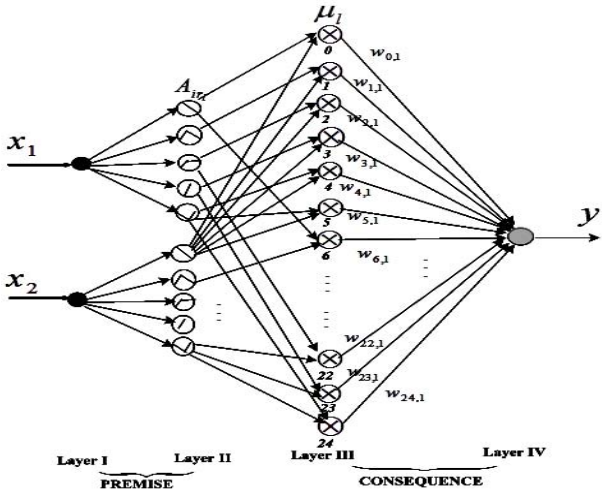


Fig. 16. Original configuration of neuro-fuzzy system with two input variables, each variable with five MFs.

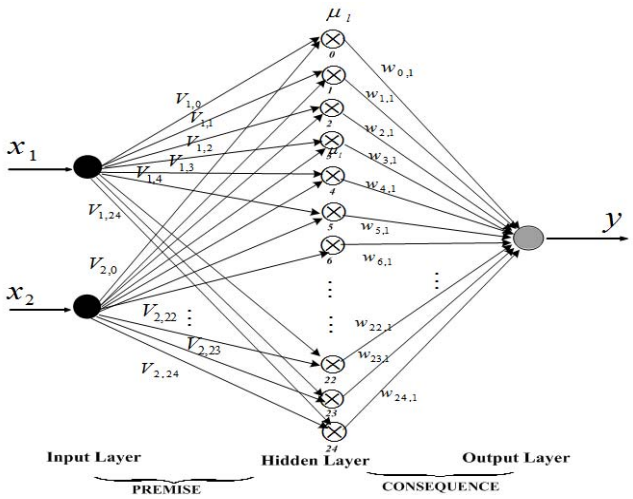


Fig. 17. F-CONFIS with two input variables, each variable with five MFs. It is good practice to explain the significance of the figure in the caption.

TABLE VII
MSE COMPARISON OF DIFFERENT ALGORITHM

Method	Epochs	MSE
Conventional gradient decent algorithm	1000	0.087277
Proposed optimal learning algorithm	200	0.077028

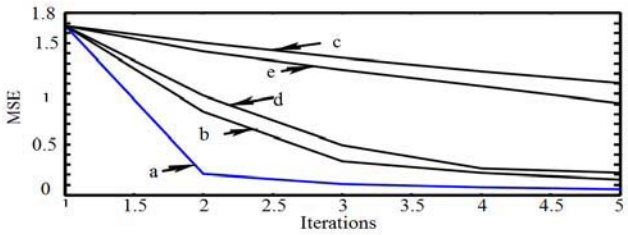


Fig. 18. Performance comparison for Example 3. Case a: proposed method. Case b: $\alpha = 0.5$, $\beta = 0.8$. Case c: $\alpha = 0$, $\beta = 0.6$. Case d: $\alpha = 0.4$, $\beta = 0.4$. Case e: $\alpha = 0.1$, $\beta = 0.2$.

Example 3 *(Gas Furnace Data With the New Algorithm):* The gas furnace data set is a commonly used benchmark. This data set has been used extensively as a benchmark example in neuro-fuzzy field [37], [38]. There are 296 input–output measurements. The input $u(t)$ is the flow rate of the methane gas and the output measurement $y(t)$ is the concentration of carbon dioxide in the gas mixture flowing out of the furnace under a steady air supply. In our experiment, the inputs are $u(t-4)$ and $y(t-1)$, with the output variable is $y(t)$. The F-CONFIS shown in Fig. 17 is equivalent to the original neuro-fuzzy system shown in Fig. 16, it has two input nodes and one output node, each input variable has five items, so there are 25 fuzzy rules. The learning rates for the conventional algorithm are set as $\alpha = 0.01$ and $\beta = 0.2$. The MFs are the Gaussian functions.

Table VII shows MSE comparison of between conventional gradient decent algorithm and the proposed method, the MSE of the conventional gradient decent algorithm reaches 0.087277 in 1000 epochs, but with the proposed dynamic optimal learning rates the MSE reaches 0.077028 in 200 epochs. The proposed dynamic learning rate approach converges much faster than the conventional gradient decent algorithm. It can be seen that the proposed tuning algorithm for F-CONFIS is very effective.

After 500 iterations, the trajectories of error of first five iterations are shown in Fig. 18. It also shows the performance comparison with other Cases b–e in which the learning rates are fixed. Table VIII shows the comparison performance of fixed learning rates and the proposed method

TABLE VIII
PERFORMANCE COMPARISON OF FIXED LEARNING RATE AND OUR METHOD

Method	α	β	MSE t=50	MSE t=200	MSE t=500
Proposed	dynamic	dynamic	0.0985	0.07702	0.07079
case b	0.5	0.8	0.2393	0.08498	0.07896
case c	0	0.6	0.9454	0.25669	0.12927
case d	0.4	0.4	0.4265	0.10249	0.08097
case e	0.1	0.2	0.7486	0.24273	0.1146

in 50, 100, 200, and 500 iterations (t). Apparently, the convergence speed of proposed algorithm is much faster than those of other cases. Fig. 19 shows the output of F-CONFIS trained by the dynamic learning rates, where the solid line is the original data and the dot denotes the F-CONFIS output with the dynamic learning rate algorithm. It is shown that the proposed approach achieves excellent performance.

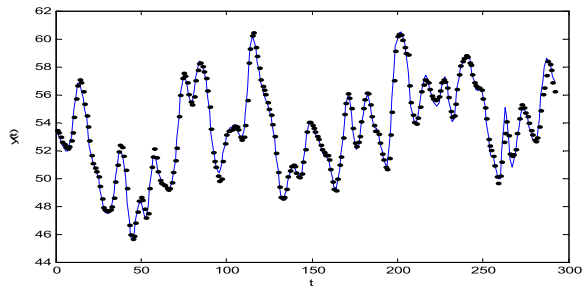


Fig. 19. Output of F-CONFIS trained by proposed method, where the solid line is the original data and the dotted line denotes the F-CONFIS outputs.

VI. Conclusion

The conventional four-layer neuro-fuzzy system is transformed into an equivalent fully connected three-layer feedforward NN, or F-CONFIS. Because F-CONFIS is a variation of multilayer NNs and has dependent and repeated links, the derivation of the learning algorithm is carefully designed to cope with weights in these repeated links. In addition, the dynamic optimal learning of F-CONFIS is derived not only in the premise part, but also in the consequent part. The simulation results indicate the proposed dynamic learning improves the accuracy considerably and converges much fast.

Appendix

The following Lemma 1 shows the increment rule of ordinal indices for hidden neurons with links of repeated weights, which is needed to prove Theorem 1.

Lemma 1: Let $\{h_\lambda(i,q)|\lambda = 1, 2, \ldots, \mathrm{RW}(i)\}$ be the ordinal indices for hidden neurons with links of repeated weights, which connect the qth MF of the ith fuzzy input variable x_i. Then, we have

$$\begin{cases} h_{\lambda+1}(i,q) = h_\lambda(i,q) + R_i s_i - s_i + 1, & \text{if } h_\lambda(i,q)\%s_i = s_i - 1 \\ h_{\lambda+1}(i,q) = h_\lambda(i,q) + 1, & \text{otherwise} \end{cases}$$

where s_i is defined in (15).

Proof: We know that $h_\lambda\ (i,\ q)$ is a specific fuzzy rule number l in (17) for which we should have $r_i(l) = q$, or

$$\begin{aligned} h_\lambda(i,q) &= l(r_1(\lambda), r_2(\lambda), \ldots, r_{i-1}(\lambda), q, r_{i+1}(\lambda), \ldots, r_N(\lambda)) \\ &= \left(\sum_{n=1}^{i-1} r_n(\lambda)s_n + qs_i + \sum_{n=i+1}^{N} r_n(\lambda)s_n\right) \\ &= \sum_{n=1}^{i-1} r_n(\lambda)s_n + qs_i \\ &\quad + R_i s_i \left(r_{i+1}(\lambda) + r_{i+2}(\lambda)R_{i+1} \right. \\ &\quad \left. + \cdots + r_N(\lambda)\prod_{n=i+1}^{N-1} R_n\right). \end{aligned} \tag{A.1}$$

Let

$$lv(\lambda) = lv\left(r_1(\lambda), r_2(\lambda), \ldots, r_{i-1}(\lambda)\right) = \sum_{n=1}^{i-1} r_n(\lambda)s_n$$

$$hv(\lambda) = hv\left(r_{i+1}(\lambda), r_{i+2}(\lambda), \ldots, r_N(\lambda)\right) = \frac{1}{R_i s_i}\sum_{n=i+1}^{N} r_n(\lambda)s_n$$

$(lv(\lambda) = 0,\ \text{if}\ \ i = 1;\ hv(\lambda) = 0,\ \text{if}\ \ i = N).$

Then, (A.1) can be rewritten as

$$h_\lambda(i,q) = lv(\lambda) + qs_i + R_i s_i hv(\lambda). \tag{A.2}$$

The (A.1) and (A.2) are actually from (17) so that all the $h_\lambda(i,q)$ in $H = \{h_\lambda(i,q)|\lambda = 1, \ldots, \mathrm{RW}(i)\}$ are arranged in ascending order by the following mechanism.

1) Let $(r_1, r_2, \ldots, r_{i-1}, r_{i+1}, \ldots, r_N) = (0, 0, \ldots, 0, 0, \ldots, 0)$.
2) Increment the $h_\lambda\ (i,\ q)$ by increasing the $(r_1,\ r_2,\ \ldots,\ r_{i-1},\ r_{i+1},\ \ldots,\ r_N)$ using the following sequence: {(0, 0, …, 0, 0, …, 0), (1, 0, …, 0, 0, …, 0), …, $(R_1 - 1$, 0, …, 0, 0, …, 0), (0, 1, …, 0, 0, …, 0), …, $(R_1 - 1$, 1, …, 0, 0, …, 0), …, $(R_1 - 1$, $R_2 - 1$, …, 0, 0, …, 0), …, $(R_1 - 1$, $R_2 - 1$, …, $R_{i-1} - 1$, $R_{i+1} - 1$, …, $R_N - 1)$}.

The reason for the above arrangement is quite obvious due to the fact that $s_{i+1} > s_i$ in (A.1), and the only way to let $h_\lambda\ (i,\ q)$ in $H = \{h_\lambda(i,q)|k = 1, \ldots, \mathrm{RW}(i)\}$ in ascending order is to increase the indices of r_i quicker than r_{i+1}.

Denote

$$\begin{aligned} &lv(r_1, r_2, \ldots, r_{i-1}) \\ &\quad = \sum_{n=1}^{i-1} r_n s_n\ , (lv(r_1, r_2, \ldots, r_{i-1}) = 0, \quad \text{if } i = 1) \\ &hv(r_{i+1}, \ldots, r_N) \\ &\quad = \frac{1}{s_{i+1}}\sum_{n=i+1}^{N} r_n s_n\ , (hv(r_1, r_2, \ldots, r_{i-1}) = 0, \quad \text{if } i = n) \end{aligned}$$

It is obvious that the minimum value of $lv(r_1, r_2, \ldots, r_{i-1})$ is

$$lv_{\min} = lv\left(0, 0, \ldots, 0\right) = \sum_{n=1}^{i-1} 0 s_n = 0.$$

The maximum value can be similarly found as

$$\begin{aligned} lv_{\max} &= lv\left(R_1 - 1, R_2 - 1, \ldots, R_{i-1} - 1\right) \\ &= \sum_{n=1}^{i-1}(R_n - 1)s_n = \sum_{n=1}^{i-1}(s_{n+1} - s_n) = s_i - s_1 = s_i - 1. \end{aligned}$$

Then, we have

$$0 \le lv(\lambda) \le s_{i-1}. \tag{A.3}$$

So, $lv(\lambda)$ may take any natural number between 0 and $s_i - 1$. The minimum value of $hv(r_{i+1}, \ldots, r_N)$ is

$$hv_{\min} = hv(0, 0, \ldots, 0) = 0 + 0R_{i+1} + \cdots + 0\prod_{n=i+1}^{N-1} R_n = 0.$$

Its maximum value can be found as

$$hv_{\max} = hv(R_{i+1}, R_{i+2}, \ldots, R_N) = \frac{1}{s_{i+1}} \sum_{n=i+1}^{N} (R_n - 1)s_n$$
$$= \frac{1}{s_{i+1}} \sum_{n=i+1}^{N} (s_{n+1} - s_n) = \frac{s_{N+1} - s_{i+1}}{s_{i+1}} = \frac{s_{N+1}}{s_{i+1}} - 1.$$

Then, we have

$$0 \le hv(\lambda) \le (s_{N+1}/s_{i+1}) - 1. \tag{A.4}$$

So, $hv(\lambda)$ may take any natural number between 0 and $(s_{N+1}/s_{i+1}) - 1$. From (A.2)–(A.4), for any natural number x in $[0, s_i - 1]$ and y in $[0, (s_{N+1}/s_{i+1}) - 1]$, we have

$$(x + qs_i + R_i s_i y) \in H = \{h_\lambda(i, q) | \lambda = 1, \ldots, \mathrm{RW}(i)\}. \tag{A.5}$$

It is also important to say that we have to adopt the above ascending order mechanism for $[xy] = [lv(r_1, r_2, \ldots, r_{i-1})hv(r_{i+1}, \ldots, r_N)]$ so that the $h_\lambda(i, q)$ in $H = \{h_\lambda(i, q) | \lambda = 1, \ldots, \mathrm{RW}(i)\}$ will be in ascending order. From (A.2) with the fact that $0 \le lv(\lambda) \le s_i - 1$, we have

$$h_\lambda(i, q)\%s_i = lv(\lambda)\%s_i + 0 + 0 = lv(\lambda). \tag{A.6}$$

The above (A.6) will be used to check if the maximum value of $lv(\lambda)$ has been reached or not. From (A.2), we have

$$h_{\lambda+1}(i, q) - h_\lambda(i, q) = lv(\lambda + 1) - lv(\lambda) + R_i s_i[hv(\lambda + 1) - hv(\lambda)]$$
$$= x - lv(\lambda) + R_i s_i[y - hv(\lambda)]. \tag{A.7}$$

Since we have assumed that all the $h_k(i, q)$ in $H = \{h_\lambda(i, q) | \lambda = 1, \ldots, \mathrm{RW}(i)\}$ are arranged in ascending order in a manner explained in the above ascending order mechanism, therefore the $x = lv(\lambda+1)$ in (A.7) will be incremented by 1 from $lv(\lambda)$ until it reaches its maximum, i.e., s_{i-1}. However, y will not be incremented in this stage, i.e., we let $yhv(\lambda) = 0$. Therefore, we have the following Case 1, i.e., if $h_\lambda(i, q)\%s_i = lv(\lambda) < s_i - 1$.

Case 1: If $h_\lambda(i, q)\%s_i = lv(\lambda) < s_i - 1$, then

$$h_{\lambda+1}(i, q) - h_\lambda(i, q) = x - lv(\lambda) + R_i s_i[y - hv(\lambda)]$$
$$= 1 + 0 = 1.$$

Then, if $h_\lambda(i, q)\%s_i = lv(\lambda) = s_i - 1$, then $x = lv(\lambda+1)$ must be reset to zero, so the $y = hv(\lambda+1)$ will be incremented from $hv(\lambda)$ by 1 until it reaches its maximum value $(s_{N+1}/s_{i+1}) - 1$. So, we have the following Case 2, i.e., if $h_\lambda(i, q)\%s_i = lv(\lambda) = s_i - 1$.

Case 2: If $h_\lambda(i, q)\%s_i = lv(\lambda) = s_i - 1$, then we reset $\boldsymbol{x}$ to zero and let $y = hv(\lambda) + 1$ to have

$$h_{\lambda+1}(i, q) - h_\lambda(i, q) = x - lv(\lambda) + R_i s_i[y - hv(\lambda)]$$
$$= 0 - lv(\lambda) + R_i s_i = R_i s_i - s_i + 1.$$

After combining the above Cases 1 and 2, we have the following conclusion:

$$\begin{cases} h_{\lambda+1}(i, q) = h_\lambda(i, q) + R_i s_i - s_i + 1, & \text{if } h_\lambda(i, q)\%s_i = s_i - 1 \\ h_{\lambda+1}(i, q) = h_\lambda(i, q) + 1, & \text{otherwise.} \end{cases}$$

Q.E.D.

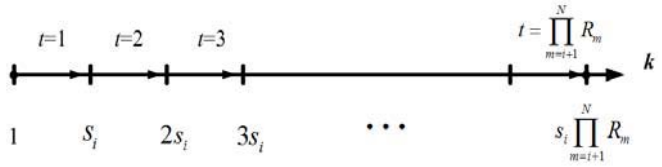


Fig. 20. Successive intervals for k with respect to t in Theorem 1.

Theorem 1: The ordinal indices $\{h_\lambda(i, q) | \lambda = 1, \ldots, \mathrm{RW}(i)\}$ for hidden neurons with links of repeated weights, which connect the qth MF of the ith fuzzy input variable x_i, can be found from the following equation:

$$h_\lambda(i, q) = (\lambda - 1)\%s_i + qs_i + [(\lambda - 1)/s_i]R_i s_i$$
$$\lambda = 1, \ldots, \mathrm{RW}(i) \tag{A.8}$$

where $A \% B$ is the remainder of the division of A over B, and A/B is the quotient of the integer division of A over B.

Proof: For the ith fuzzy input variable x_i, by (3), we have

$$\mathrm{RW}(i) = \frac{\prod_{m=1}^{N} R_m}{R_i} = \frac{\left(\prod_{m=1}^{i-1} R_m\right) R_i \left(\prod_{m=i+1}^{N} R_m\right)}{R_i}$$
$$= s_i \prod_{m=i+1}^{N} R_m.$$

Therefore, we have $1 \le \lambda \le s_i \prod_{m=i+1}^{N} R_m$, which will fall into one of the ranges $\{[(t-1)s_i + 1, ts_i] | \ t = 1, \ldots, \prod_{m=i+1}^{N} R_m\}$ (if $i = N$, then $t = 1$ and k ranges from 1 to s_i). This can be illustrated by Fig. 20.

If we can prove (A.8) holds for all t, then it accordingly holds for all λ. We will prove this by mathematical induction.

Base: When $t = 1$, λ will fall into the range of $[1, s_i]$. For $\lambda = 1$, from (A.2) with the fact that $\{r_1(1)r_2(1)\ldots r_{i-1}(1)r_{i+1}(1)\ldots sr_N(1)\} = \{00\ldots00\ldots0\}$

$$h_{\lambda=1}(i, q) = 0 + qs_i + 0 = 0 + qs_i + (\lambda - 1).$$

This will imply $h_1(i, q)\%s_i = 0$. Therefore, according to Lemma 1, $h_2(i, q) = h_1(i, q) + 1 = qs_i + 1$. This process can be continued as follows:

$$\begin{aligned} h_1(i, q)\%s_i &= 0 \Rightarrow h_2(i, q) = h_1(i, q) + 1 = qs_i + 1; \\ h_2(i, q)\%s_i &= 1 \Rightarrow h_3(i, q) = h_2(i, q) + 1 = qs_i + 2; \\ &\ldots \\ h_{s_i-2}(i, q)\%s_i &= s_i - 3 \Rightarrow h_{s_i-1}(i, q) = h_{s_i-2}(i, q) + 1 \\ &= qs_i + s_i - 2; \\ h_{s_i-1}(i, q)\%s_i &= s_i - 2 \Rightarrow h_{s_i}(i, q) = h_{s_i-1}(i, q) + 1 \\ &= qs_i + s_i - 1. \end{aligned}$$

So that we have $h_\lambda(i, q)\%s_i = \lambda - 1, if 1 \le \lambda \le s_i$ and

$$h_\lambda(i, q) = qs_i + \lambda - 1, \quad 1 \le \lambda \le s_i. \tag{A.9}$$

For $1 \leq \lambda \leq s_i$, we have $(\lambda - 1)/s_i = 0$ and $(\lambda - 1) = (\lambda - 1)\%s_i$. From (A.9), we have

$$h_\lambda(i,q) = qs_i + \lambda - 1 = [(\lambda-1)/s_i]R_i s_i + qs_i + (\lambda-1)\%s_i, \quad 1 \leq \lambda \leq s_i.$$

So (A.8) holds for $t = 1$.

Induction Hypothesis: Assume (A.8) holds for some specified t_0, then we have

$$h_\lambda(i,q) = [(\lambda-1)/s_i]R_i s_i + qs_i + (\lambda-1)\%s_i \quad (t_0-1)s_i + 1 \leq \lambda \leq t_0 s_i. \tag{A.10}$$

Inductive Step: The last value of $\{h_\lambda(i,q) |\ (t_0-1)s_i + 1 \leq \lambda \leq t_0 s_i\}$ is

$$\begin{aligned} h_{\lambda=t_0 s_i}(i,q) &= [(t_0 s_i - 1)/s_i]R_i s_i + qs_i + (t_0 s_i - 1)\%s_i \\ &= [((t_0-1)s_i + s_i - 1)/s_i]R_i s_i + qs_i \\ &\quad + [(t_0-1)s_i + s_i - 1]\%s_i \\ &= (t_0-1)R_i s_i + qs_i + s_i - 1. \end{aligned} \tag{A.11}$$

From (A.11), we have $h_{\lambda=t_0 s_i}(i,q)\%s_i = s_i - 1$. According to the Lemma 1, we have

$$\begin{aligned} h_{\lambda=t_0 s_i+1}(i,q) &= h_{\lambda=t_0 s_i}(i,q) + R_i s_i - s_i + 1 \\ &= (t_0-1)R_i s_i + qs_i + s_i - 1 + R_i s_i - s_i + 1 \\ &= t_0 R_i s_i + qs_i. \end{aligned} \tag{A.12}$$

For $\lambda = ts_i + 1$, we have $(\lambda-1)/s_i = t_0$ and $(\lambda-1)\%s_i = 0$. From (A.12), we have

$$\begin{aligned} h_{\lambda=t_0 s_i+1}(i,q) &= t_0 R_i s_i + qs_i \\ &= [(\lambda-1)/s_i]R_i s_i + qs_i \\ &\quad + (\lambda-1)\%s_i. \end{aligned} \tag{A.13}$$

From (A.13), $h_{\lambda=t_0 s_i+1}(i,q)\%s_i = 0$. Therefore, the next $s_i - 1$ items of $\{h_\lambda(i,q) | t_0 s_i + 2 \leq \lambda \leq (t_0+1)s_i\}$ will have the property of $\{h_\lambda(i,q)\%s_i < s_i - 1 | t_0 s_i + 2 \leq \lambda \leq (t_0+1)s_i\}$. According to Lemma 1, $h_{\lambda+1}(i,q) = h_\lambda(i,q) + 1$. Therefore, we have

$$h_\lambda(i,q) = t_0 R_i s_i + qs_i + (\lambda-1)\%s_i \quad t_0 s_i + 2 \leq \lambda \leq (t_0+1)s_i. \tag{A.14}$$

For $t_0 s_i + 1 \leq \lambda \leq (t_0+1)s_i$, we have $(\lambda-1)/s_i = t_0$. Therefore, we can combine (A.13) and (A.14) to have

$$h_\lambda(i,q) = [(\lambda-1)/s_i]\, R_i s_i + qs_i + (\lambda-1)\%s_i \quad t_0 s_i + 1 \leq \lambda \leq (t_0+1)s_i.$$

Thereby, (A.8) holds for $t_0 + 1$.

Q.E.D.

REFERENCES

[1] A. Miranian and M. Abdollahzade, "Developing a local least-squares support vector machines-based neuro-fuzzy model for nonlinear and chaotic time series prediction," *IEEE Trans. Neural Netw.*, vol. 24, no. 2, pp. 207–218, Feb. 2013.

[2] E. Y. Cheu, C. Quek, and S. K. Ng, "ARPOP: An appetitive reward-based pseudo-outer-product neural fuzzy inference system inspired from the operant conditioning of feeding behavior in Aplysia," *IEEE Trans. Neural Netw.*, vol. 23, no. 2, pp. 317–328, Feb. 2012.

[3] C. H. Wang, W. Y. Wang, T. T. Lee, and P.-S. Tseng, "Fuzzy B-spline membership function and its applications in fuzzy-neural control," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, no. 5, pp. 841–851, May 1995.

[4] C. L. P. Chen and Y. H. Pao, "An integration of neural network and rule-based systems for design and planning of mechanical assemblies," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, no. 5, pp. 1359–1371, Sep./Oct. 1993.

[5] P. Melin and O. Castillo, "Adaptive intelligent control of aircraft systems with a hybrid approach combining neural networks, fuzzy logic and fractal theory," *Appl. Soft Comput.*, vol. 3, no. 4, pp. 353–362, Dec. 2003.

[6] Z. Deng, Y. Jiang, K. S. Choi, F. L. Chung, and S. Wang, "Knowledge-leverage-based TSK fuzzy system modeling," *IEEE Trans. Neural Netw.*, vol. 24, no. 8, pp. 1200–1212, Aug. 2013.

[7] C. Y. Yeh, W. H. R. Jeng, and S. J. Lee, "Data-Based system modeling using a type-2 fuzzy neural network with a hybrid learning algorithm," *IEEE Trans. Neural Netw.*, vol. 22, no. 12, pp. 2296–2309, Dec. 2011.

[8] J. S. R. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, no. 3, pp. 665–685, May 1993.

[9] H. X. Li and C. L. P. Chen, "The equivalence between fuzzy logic systems and feedforward neural networks," *IEEE Trans. Neural Netw.*, vol. 11, no. 2, pp. 356–365, Mar. 2000.

[10] L. Tesli´c, B. Hartmann, O. Nelles, and I. Škrjanc, "Nonlinear system identification by gustafson-kessel fuzzy clustering and supervised local model network learning for the drug absorption spectra process," *IEEE Trans. Neural Netw.*, vol. 22, no. 12, pp. 1941–1951, Dec. 2011.

[11] X. H. Yu, "Can back-propagation error surface not have local minima," *IEEE Trans. Neural Netw.*, vol. 3, no. 6, pp. 1019–1021, Nov. 1992.

[12] X. H. Yu, G.-A. Chen, and S. Xin, "Dynamic learning rate optimization of the backpropagation algorithm," *IEEE Trans. Neural Netw.*, vol. 6, no. 3, pp. 669–677, May 1995.

[13] P. Melin and O. Castillo, "Intelligent control of a stepping motor drive using an adaptive neuro-fuzzy inference system," *Inf. Sci.*, vol. 170, nos. 2–4, pp. 133–151, Feb. 2005.

[14] L. Aguilar, P. Melin, and O. Castillo, "Intelligent control of a stepping motor drive using a hybrid neuro-fuzzy ANFIS approach," *Appl. Soft Comput.*, vol. 3, no. 3, pp. 209–219, Nov. 2003.

[15] D. S. Chen and R. C. Jain, "A robust back propagation learning algorithm for function approximation," *IEEE Trans. Neural Netw.*, vol. 5, no. 3, pp. 467–479, May 1994.

[16] K. C. Tan and H. J. Tang, "New dynamical optimal learning for linear multilayer FNN," *IEEE Trans. Neural Netw.*, vol. 15, no. 6, pp. 1562–1570, Nov. 2004.

[17] J. Wang, C. H. Wang, and C. L. Philip Chen, "The bounded capacity of fuzzy neural networks (FNNs) via a new fully connected neural fuzzy iference system (F-CONFIS) with Its applications," *IEEE Trans. Fuzzy Syst.*, doi 10.1109/TFUZZ.2013.2292972, 2014.

[18] C. H. Wang, H. L. Liu, and C. T. Lin, "Dynamic optimal learning rates of a certain class of fuzzy neural networks and its applications with genetic algorithm," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 31, no. 3, pp. 467–475, Jun. 2001.

[19] G. D. Magoulas, M. N. Vrahatis, and G. S. Androulakis, "Effective back-propagation with variable stepsize," *Neural Netw.*, vol. 10, no. 1, pp. 69–82, 1997.

[20] W. Wu, L. Li, J. Yang, and Y. Liu, "A modified gradient-based neuro-fuzzy learning algorithm and its convergence," *Inf. Sci.*, vol. 180, no. 9, pp. 1630–1642, May 2010.

[21] C. H. Wang, C. S. Cheng, and T. T. Lee, "Dynamical optimal learning for FNN and its applications," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 3, pp. 467–475, Jun. 2004.

[22] H. J. Tang, K. C. Tan, and T. H. Lee, "Dynamical optimal learning for FNN and its applications," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, vol. 1. Jul. 2004, pp. 443–447.

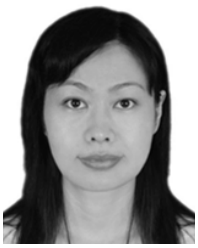
[23] J. S. R. Jang, C. T. Sun, and E. Mizutani, *Neuro-fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Upper Saddle River, NJ, USA: Prentice-Hall, 1997.

[24] C. H. Wang and J. S. Wen, "On the equivalence of a table lookup (TL)technique and fuzzy neural network (FNN) with block pulse membership functions (BPMFs) and its application to water injection control of an automobile," *IEEE Trans Syst., Man, Cybern. C, Appl. Rev.*, vol. 38, no. 4, pp. 574–580, Jul. 2008.
[25] C. C. Chuang, S. F. Su, and C. C. Hsiao, "The annealing robust backpropagation (ARBP) learning algorithm," *IEEE Trans. Neural Netw.*, vol. 11, no. 5, pp. 1067–1076, Sep. 2000.
[26] M. T. Hagan, H. B. Demuth, and M. Beale, *Neural Network Design*. Beijing, China: Machine Press, 2002.
[27] A. K. Palit and G. Doeding, "Backpropagation based training algorithm for takagi-sugeno type MIMO neuro-fuzzy network to forecast electrical load time series," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, vol. 1, May 2002, pp. 86–91.
[28] M. N. H. Siddique and M. O. Tokhi, "Training neural networks: Backpropagation vs. genetic algorithms," in *Proc. Int. Joint Conf. Neural Netw.*, vol. 4. Jul. 2001. pp. 2673–2678.
[29] J. Nocedal and S. J. Wright, *Numerical Optimization. Second Edition*. New York, NY, USA: Springer-Verlag, 2006.
[30] J. L. Aznarte and J. M. Benítez, "Equivalences between neural-autoregressive time series models and fuzzy systems," *IEEE Trans. Neural Netw.*, vol. 21, no. 9, pp. 1434–1444, Sep. 2010.
[31] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York, NY, USA: Springer-Verlag, 2006.
[32] L. Armijo, "Minimization of functions having Lipschitz continuous partial derivatives," *Pacific J. Math.*, vol. 16, no. 1, pp. 1–4, 1966.
[33] F. A. Potra and Y. Shi, "Efficient line search algorithm for unconstrained optimization," *J. Optim. Theory Appl.*, vol. 85, no. 3, pp. 677–704, 1995.
[34] W. Q. Zhao, K. Li, and G. W. Irwin, "A new gradient descent approach for local learning of fuzzy neural models," *IEEE Trans. Fuzzy Syst.*, vol. 21, no. 1, pp. 30–44, Feb. 2013.
[35] C. S. Ouyang, W. J. Lee, and S. J. Lee, "A TSK-type neuro fuzzy network approach to system modeling problems," *IEEE Trans. Syst., Man, Cybern.*, vol. 35, no. 4, pp. 751–767, Aug. 2005.
[36] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Netw.*, vol. 1, no. 1, pp. 4–26, Mar. 1990.
[37] R. Jang, "ANFIS: Adaptive network-based fuzzy inference system," *IEEE Trans. Syst., Man, Cybern.* vol. 23, no. 3, pp. 665–685, May/Jun. 1993.
[38] J. Zhao, W. Wang, Y. Liu, and W. Pedrycz, "A two-stage online prediction method for a blast furnace gas system and its application," *IEEE Trans. Control Syst. Technol.*, vol. 19, no. 3, pp. 507–520, May 2011.

C. L. Philip Chen (S'88–M'88–SM'94–F'07) received the M.S. and Ph.D. degrees in electrical engineering from the University of Michigan, Ann Arbor, MI, USA, and the Purdue University, West Lafayette, IN, USA, in 1985 and 1988, respectively.

He is currently a Chair Professor with the Department of Computer and Information Science and the Dean with the Faculty of Science and Technology, University of Macau, Macau, China, after having worked at U.S. for 23 years as a Tenured Professor, the Department Head, and the Associate Dean in two different universities. His current research interests include computational intelligence, systems, and cybernetics.

Dr. Chen is a fellow of American Association for the Advancement of Science and HKIE. He is currently the Junior President of the IEEE Systems, Man, and Cybernetics Society.



Jing Wang (S'11) received the M.S. degree in computer science from the Guangdong University of Technology, Guangzhou, China, in 2007. She is currently pursuing the Ph.D. degree in software engineering with the Faculty of Science and Technology, University of Macau, Macau, China.

Her current research interests include neuro-fuzzy system, neural network, intelligent control, cyber physical system, machine learning techniques and their applications, computational intelligent, systems, and cybernetics.

Chi-Hsu Wang (M'92–SM'93–F'08) received the B.S. degree in control engineering from the National Chiao Tung University, Hsinchu, Taiwan, the M.S. degree in computer science from the National Tsing Hua University, Hsinchu, and the Ph.D. degree in electrical and computer engineering from the University of Wisconsin, Madison, WI, USA, in 1976, 1978, and 1986, respectively.

He was an Associate Professor in 1986 and a Professor in 1990 with the Department of Electrical Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan. He is currently a Professor with the Department of Electrical Engineering, National Chiao Tung University. He has been the Adjunct Professor with North East University, Qinhuangdao, China, since 2014. His current research interests and publications include in the areas of digital control, fuzzy-neural-network, intelligent control, adaptive control, and robotics.

Long Chen (M'11) received the B.S. degree in information sciences from Peking University, Beijing, China, the M.S.E. degree from the Institute of Automation, Chinese Academy of Sciences, Beijing, the M.S. degree in computer engineering from the University of Alberta, Edmonton, AB, Canada, and the Ph.D. degree in electrical engineering from the University of Texas at San Antonio, TX, USA, in 2000, 2003, 2005, and 2010, respectively.

He was a Post-Doctoral Fellow with the University of Texas at San Antonio from 2010 to 2011. He is currently an Assistant Professor with the Department of Computer and Information Science, University of Macau, Macau, China. His current research interests include computational intelligence, Bayesian methods, and other machine learning techniques and their applications.

Mr. Chen has been working in publication matters for many IEEE conferences and is the Publications Co-Chair of the 2009 IEEE International Conference on Systems, Man, and Cybernetics.