Applying Fuzzy Adaptive Network
to Fuzzy Regression Analysis

C.-B. CHENG AND E. S. LEE*
Department of Industrial and Manufacturing Systems Engineering
Kansas State University, Manhattan, KS 66506, U.S.A.

(Received and accepted June 1998)

Abstract—A fuzzy inference system based on the Sugeno inference model is first formulated for fuzzy regression analysis. This system is then represented by a fuzzy adaptive network. This approach combines the power of representation of fuzzy inference system with the ability of learning of the neural network. Numerical examples are trained and solved to illustrate the approach. The results are compared with other approaches. © 1999 Elsevier Science Ltd. All rights reserved.

Keywords—Fuzzy regression, Fuzzy adaptive network, Fuzzy inference system, Learning rules.

1. INTRODUCTION

Neural network based on the back propagation approach has been applied to nonparametric regression by several investigators. According to the connection weights used, they can be classified as:

(a) nonfuzzy weights: the investigations by Ishibuchi and Tanaka [1] and Fedrizzi et al. [2];
(b) symmetric fuzzy weights: this category includes the works of Ishibuchi et al. [3,4] and Miyazaki et al. [5]; and
(c) nonsymmetric fuzzy weights: the works of Ishibuchi and Nii [6] and Ishibuchi et al. [7].

Although these approaches are fairly powerful, a combination of fuzzy inference with neural network learning for nonparametric regression should form an even more powerful approach. However, very little had been done in this direction. The purpose of this paper is to propose and investigate such a system, namely, the Fuzzy Adaptive Network (FAN).

Cheng and Lee [8] have proposed a Fuzzy Radial Basis Function Network (FRBFN) for fuzzy regression analysis. Jang and Sun [9] have pointed out that the crisp radial basis function network is functional equivalent to the fuzzy inference system. Thus, another purpose of this paper is to compare the FRBFN network with the proposed fuzzy adaptive network.

Pokorný [10] has applied the Sugeno fuzzy inference system [11] to fuzzy nonlinear regression. In Pokorný's model, the crisp linear functions in the consequence section of the Sugeno fuzzy inference system are replaced by possibilistic linear equations. The training of this model is similar to the original Sugeno fuzzy model, but the updating of the consequence section is formulated as a possibilistic linear model instead of a least square one.

*Author to whom all correspondence should be addressed.
In this paper, we adopt the idea of Pokorný, but instead of directly applying the fuzzy inference system to fuzzy regression, we represent the resulting inference system by a fuzzy adaptive network. Through this adaptive network, the learning algorithms which are developed for neural networks can be used for the proposed inference system.

The training of FAN is consisted of two parts:

(1) the identification of the premise parameters—this identification is carried out by the back propagation algorithm; and

(2) the identification of the consequence parameters—this identification is carried out by solving a possibilistic linear system.

To illustrate the approach, numerical examples are solved and the results are compared with the FRBFN approach.

2. FUZZY INFERENCE SYSTEM

The fuzzy inference system forms a useful computing framework based on the concepts of fuzzy set theory, fuzzy if-then rules, and fuzzy reasoning. This system has been applied successfully in various fields such as automatic control, expert systems, computer vision, etc. The fuzzy inference system is a powerful function approximator, and it differs from another powerful function approximator, neural networks, in its capability of handling linguistic information.

The basic structure of a fuzzy inference system consists of three conceptual components [9]:

(1) a rule base, which contains a selection of fuzzy rules;

(2) a database, which defines the membership functions used in the fuzzy rules; and

(3) a reasoning mechanism, which performs the inference procedure upon the rules to derive a reasonable output.

The fuzzy rule or usually called fuzzy if-then rule is of the following form:

\[ R : \text{if } (x_1 \text{ is } F_1, \text{ and } x_2 \text{ is } F_2 \ldots \text{ and } x_p \text{ is } F_p), \text{ then } Y \text{ is } G, \]

where \( F_i, i = 1, \ldots, p, \) and \( G \) are linguistic terms which are fuzzy sets defined by membership functions, and \( X = (x_1, \ldots, x_p)^T \) and \( Y \) are the input and output linguistic variables, respectively. The statement in the antecedent or premise represents the input information and the statement in the consequence or conclusion represents the output. If-then statements are used in almost all of our daily activities. For example:

if the speed is high, then apply the brake a little;

if the temperature is very hot, then turn the air conditioner higher;

where "speed" and "temperature" are linguistic variables and "high" and "very hot" are linguistic terms. It is obviously that "high" and "hot" are fuzzy. We need to define membership functions for these fuzzy linguistic terms. Furthermore, to derive conclusions from a set of fuzzy if-then rules, we need an inference procedure, which is called fuzzy reasoning or approximate reasoning. These reasoning procedures derive conclusion based on information aggregation from all the rules.

Different types of fuzzy if-then rules and aggregation methods lead to different fuzzy inference systems. There are several different types of fuzzy inference systems developed for function approximation. The fuzzy inference system proposed by Takagi and Sugeno [11], which is known as the Sugeno fuzzy model or Sugeno fuzzy inference system will be used in the present investigation. Instead of the if-then rules listed in equation (1), Takagi and Sugeno proposed the following fuzzy rule:

\[ R^t : \text{if } (x_1 \text{ is } F_1^t, \text{ and } x_2 \text{ is } F_2^t \ldots \text{ and } x_p \text{ is } F_p^t), \text{ then } Y = Y^t = c_0^t + c_1^t x_1 + \cdots + c_p^t x_p, \]

where \( c_0^t, c_1^t, \ldots, c_p^t \) are the consequence parameters which are determined by solving a possibilistic linear system.
in which $F^i_l$ represents fuzzy set or fuzzy terms associated with the input $x_i$ in the $l^{th}$ rule, $Y^i_l$ is the system output due to rule $R^l$, and there are $m$ rules, $l = 1, 2, \ldots, m$. In the Sugeno fuzzy system, $c^i_l$ represents real-valued parameter. In the present application, $c^i_l$ will be assumed to be a fuzzy number so that fuzzy output can be obtained in the regression analysis. Thus, the consequence in (2) is a possibilistic linear equation. For a real-valued input vector $X = (x_1, \ldots, x_p)^T$, the overall output of the Sugeno fuzzy system is a weighted average of the $Y^i_l$s

$$\hat{Y} = \frac{\sum_{l=1}^{m} w^l Y^l}{\sum_{l=1}^{m} w^l},$$

(3)

where the weight $w^l$ implies the truth value of the proposition $Y = Y^l$ and is defined as

$$w^l = \prod_{i=1}^{p} \mu_{F^i_l}(x_i),$$

(4)

where $\mu_{F^i_l}(x_i)$ is a membership function defined on the fuzzy set $F^i_l$. In equation (4), $w^l$ is defined in terms of a “product” operator on the membership functions. $w^l$ can also be defined differently such as the “min” operator. The advantage of the Sugeno inference system is that it provides a compact system equation and thus, parameter estimation methods can be developed easily to estimate the parameters $c^i_l$.

Fuzzy regression carried out by the use of the above inference system can be viewed from the standpoint of fuzzy partition. The premise of the fuzzy if-then rule represents the description of the fuzzy subspace of the input variables. In other words, a fuzzy partition of the input variable space is first carried out and each fuzzy subspace forms a linear input and a linear regression output. Finally, by aggregating the outputs from all the rules, a nonparametric fuzzy regression model for the original problem is obtained.

### 3. FUZZY ADAPTIVE NETWORK

We have discussed how to employ the fuzzy inference system for fuzzy regression analysis in the previous section. However, there is still a need for an effective method which can be used to fine tune or training the various parameters and the membership functions. This can be accomplished by the use of a fuzzy adaptive network FAN, which essentially is a network representation of the formulated fuzzy system. The fuzzy adaptive network based on the architecture of ANFIS proposed by Jang [12] will be used in this investigation. There are several advantages of using FAN. The network provides a comprehensive visualization of the system, and furthermore, the various learning algorithms developed for neural network and other adaptive systems can be applied based on this network. Through this learning or update approach, a good approximation of the regression function can be obtained.

The adaptive network is composed of nodes inter-connected through directional links. Part of or all the nodes are adaptive. Adaptive nodes are nodes containing parameters and the values of these parameters can be adjusted by learning.

FAN is a five-layered feed forward network in which each node performs a particular node function on the incoming signals. This node function is characterized by a set of parameters. To reflect different adaptive capabilities, the nodes are represented by circles or squares. Square nodes represent adaptive nodes with parameters and circle nodes represent fixed nodes without parameters. To illustrate how a fuzzy inference system can be represented by FAN, let us consider the following example.
Suppose a fuzzy inference system contains the following four rules:

\[ R^1 : \text{if (} x_1 \text{ is small and } x_2 \text{ is low), then } (Y = Y^1 = c_0 + c_1^1 x_1 + c_2^1 x_2), \]
\[ R^2 : \text{if (} x_1 \text{ is small and } x_2 \text{ is high), then } (Y = Y^2 = c_0^2 + c_1^2 x_1 + c_2^2 x_2), \]
\[ R^3 : \text{if (} x_1 \text{ is large and } x_2 \text{ is low), then } (Y = Y^3 = c_0^3 + c_1^3 x_1 + c_2^3 x_2), \]
\[ R^4 : \text{if (} x_1 \text{ is large and } x_2 \text{ is high), then } (Y = Y^4 = c_0^4 + c_1^4 x_1 + c_2^4 x_2). \]

This system has a two-dimensional input, \( X = (x_1, x_2)^T \). For input \( x_1 \), there are two fuzzy sets "small" and "large" associated with it and for input \( x_2 \), two fuzzy sets "low" and "high" are associated with it. This fuzzy system is represented by the FAN network as shown in Figure 1.

![Figure 1. Architecture of FAN for the illustrative example.](image)

There are two subgroups of nodes in Layer 1 (see Figure 1). The first subgroup includes nodes "small" and "large", which are linked by \( x_1 \); and the second subgroup includes nodes "low" and "high", which are linked by \( x_2 \). Each node in Layer 1 outputs a membership function based on the linguistic value of the input. Nodes in Layer 2 output the products which are \( w^l, l = 1, \ldots, 4 \), based on the incoming signals. The function of a node in this layer is to synthesize the information in the premise section of the fuzzy if-then rule. For example, node 1 in Layer 2, \( A_1 \), receives signals from "small" and "low", which is equivalent to the premise of \( R^1 \) in the above fuzzy inference system. The number of nodes in Layer 2 is the number of combinations of nodes from each subgroup in Layer 1. Layer 3 simply performs a normalization of the output signals from Layer 2. Each node in Layer 4 corresponds to the consequence of each fuzzy if-then rule. For example, the first node \( Y^1 \) in Layer 4 is defined as \( Y^1 = c_0^1 + c_1^1 x_1 + c_2^1 x_2 \). Finally, Layer 5 sums up all the outputs from Layer 4, which is equivalent to perform an aggregation of all the four fuzzy if-then rules.

Let the output of node \( h \) in layer \( r \) be denoted as \( f_{r,h} \), then the functions of each node in Figures 1 can be described as follows.

**Layer 1.** Let the fuzzy sets, "small", "large", "low", and "high", in the premise section of fuzzy if-then rules be denote by \( F_1 \), \( F_2 \), \( F_3 \), and \( F_4 \), respectively. Nodes in this layer are adaptive and the output of node \( h \) is defined by the membership function on \( F_h \)

\[ f_{1,h} = \mu_{F_h}(x_1), \quad \text{for } h = 1, 2 \]
and
\[ f_{1,h} = \mu_{F_h}(x_2), \quad \text{for } h = 3, 4. \] (7)

The membership function for \( F_h \) can be any appropriated function. In this investigation, we assume a Gaussian function whose parameters can be represented by the parameter set \( \{v_h, \sigma_h\} \)
\[ \mu_{F_h}(x_1) = \exp \left[ -\left( \frac{x_1 - v_h}{\sigma_h} \right)^2 \right], \quad \text{for } h = 1, 2 \] (8)
and
\[ \mu_{F_h}(x_2) = \exp \left[ -\left( \frac{x_2 - v_h}{\sigma_h} \right)^2 \right], \quad \text{for } h = 3, 4. \] (9)

The parameter set \( \{v_h, \sigma_h\} \) in this layer is referred to as the premise parameters.

**Layer 2.** Every node in this layer is a fixed node labeled \( A_l, l = 1, \ldots, 4 \). The nodes in this layer act as the fuzzy and operator in the premise section of the fuzzy if-then rule. Each node has exactly two incoming signals from Layer 1. In this investigation, \( A_l \) is defined as a multiplication of its incoming signals. This multiplied output forms the firing strength \( w^l \) for rule \( l \):
\[ f_{2,1} = w^1 = \mu_{F_1}(x_1) \cdot \mu_{F_2}(x_2), \] (10)
\[ f_{2,2} = w^2 = \mu_{F_1}(x_1) \cdot \mu_{F_3}(x_2), \] (11)
\[ f_{2,3} = w^3 = \mu_{F_2}(x_1) \cdot \mu_{F_3}(x_2), \] (12)
and
\[ f_{2,4} = w^4 = \mu_{F_2}(x_1) \cdot \mu_{F_4}(x_2). \] (13)

**Layer 3.** Nodes in this layer are fixed nodes labeled \( N_l, l = 1, \ldots, 4 \). The output of this layer is a normalization of the outputs of Layer 2:
\[ f_{3,l} = \bar{w}^l = \frac{w^l}{\sum_{t=1}^{m} w^t}, \quad l = 1, \ldots, 4. \] (14)

**Layer 4.** The nodes in this layer are adaptive nodes with nodes function
\[ f_{4,l} = \bar{w}^l Y^l, \quad l = 1, \ldots, 4, \] (15)
where \( Y^l \) is the consequent part of a fuzzy if-then rule, and
\[ Y^l = c_{0}^l + c_{1}^l x_1 + c_{2}^l x_2, \] (16)
where \( c_{i}^l \) are fuzzy numbers and are referred to as the consequence parameters.

**Layer 5.** The single node in this layer is a fixed node, which computes the overall output as the summation of all the incoming signals
\[ f_{5,1} = \bar{Y} = \sum_{l=1}^{4} \bar{w}^l Y^l. \] (17)
4. THE TRAINING OF THE FAN

The objective of FAN is to obtain a desired nonlinear mapping between the given input-output data pairs. This desired mapping is obtained by a learning algorithm. However, in order to measure the performance of such an adaptive network, some performance or error measure is needed. In this section, the error measure will be defined first and then introduce the learning algorithm.

4.1. The Error Measure

The error measure is defined as the difference between the network or estimated outputs and the desired or target outputs. When this error measure is less than a predefined small allowed error, the training of the network is terminated. The difference between the estimated and the target outputs for each individual observation is

\[ \varepsilon_k = Y_k \{ \} \hat{Y}_k, \]

where \{\} is an operator, \( Y_k \) is the \( k \)th target output and \( \hat{Y}_k \) is the network output of the \( k \)th input vector, which is \( X_k = (x_{1k}, \ldots, x_{pk})^T \). We shall assume that both \( Y_k \) and \( \hat{Y}_k \) are symmetric triangular fuzzy numbers and can be represented by \( Y_k = (y_k, e_k) \) and \( \hat{Y}_k = (\hat{y}_k, \hat{e}_k) \), respectively. \( y_k \) and \( \hat{y}_k \) are the modes or the centers of \( Y_k \) and \( \hat{Y}_k \), and \( e_k \) and \( \hat{e}_k \) are the spreads. Assume that the consequence parameter \( c_i \) is also a symmetric triangular fuzzy number and is represented by

\[ c_i = (a_i, b_i^i), \quad i = 0, \ldots, p, \quad l = 1, \ldots, m. \]  

From the above definitions and using fuzzy arithmetic, \( \hat{y}_k \) and \( \hat{e}_k \) can be expressed as

\[ \hat{y}_k = \sum_{l=1}^{m} \sum_{i=0}^{p} \frac{a_i}{\overline{w}} x_{lk}, \]  

and

\[ \hat{e}_k = \sum_{l=1}^{m} \sum_{i=0}^{p} \frac{b_i}{\overline{w}} x_{lk}, \]  

where \( x_{0k} = 1 \).

In order to obtain the difference between two fuzzy numbers, some fuzzy ranking method must be used to define the operator \{\} in equation (18). This is because fuzzy numbers are sets, not crisp numbers. There are many fuzzy ranking methods for measuring the difference between two or more fuzzy numbers. Chang and Lee [13] made an extensive survey about fuzzy ranking methods. In the present investigation, the method of Chang and Lee [14], which is based on the concept of overall existence, will be used. An overall existence measurement of a \( L-R \) type fuzzy number \( T \) is defined as

\[ OM(T) = \int_0^1 \varpi(\nu) \left[ \chi_1(\nu)\mu_{T_{l}}^{-1}(\nu) + \chi_2(\nu)\mu_{T_{r}}^{-1}(\nu) \right] d\nu, \]  

where, \( \nu \) is the membership function; \( \mu_{T_{l}}^{-1}(\nu) \) and \( \mu_{T_{r}}^{-1}(\nu) \) are the lower and upper limits of the \( \nu \)-level set of fuzzy number \( T \); and \( \varpi(\nu), \chi_1(\nu), \) and \( \chi_2(\nu) \) are weight measures, which must be determined subjectively by the decision maker. For simplicity, we let

\[ \varpi(\nu) = 1, \quad \chi_1(\nu) = \chi_2(\nu) = \frac{1}{2}, \quad \text{for all } \nu \in (0, 1]. \]  

If \( T = (t, s^L, s^R)_{LR} \) is a triangular fuzzy number, we have

\[ OM(T) = \frac{(4t - s^L + s^R)}{4}. \]
The individual difference $\varepsilon_k$ can now be calculated by using the above equation. The average of the sum of square of $\varepsilon_k$ is then used as a measure of the overall difference. It must be noted that, due to the assumption of indifference in weights in equation (23), $\varepsilon_k$ determined by the above fuzzy ranking method does not provide a measure of the difference between the spreads of the network outputs and that of the target outputs [15]. Thus, another measure is added to calculate the difference between this spreads. Accordingly, the final overall error function is

$$E = \frac{1}{N} \left( \sum_{k=1}^{N} (y_k - \tilde{y}_k)^2 + (\varepsilon_k - \tilde{\varepsilon}_k)^2 \right),$$  

where $N$ is the number of pairs of the training data. The training of FAN terminates when $E$ is smaller than a prespecified small number.

4.2. The Learning Algorithm

Different methods were used for the training of the premise and the consequent parameters. Back propagation is used for the former and possibilistic linear programming is used to train the latter.

4.2.1. Consequence parameters

From the node functions in FAN or from equation (17), we can see that when the values of the premise parameters are fixed, the overall output can be expressed as a linear combination of the consequent parameters. Thus, according to equations (16) and (17), the output $\tilde{Y}$ can be expressed as

$$\tilde{Y} = \tilde{w}^1 (c_0^1 + c_1^1 x_1 + \cdots + c_p^1 x_p) + \tilde{w}^2 (c_0^2 + c_1^2 x_1 + \cdots + c_p^2 x_p)$$

$$+ \cdots + \tilde{w}^m (c_0^m + c_1^m x_1 + \cdots + c_p^m x_p)$$

$$= c_0^1 \tilde{w}^1 + c_1^1 (\tilde{w}^1 x_1) + \cdots + c_p^1 (\tilde{w}^1 x_p) + \cdots + c_p^m (\tilde{w}^m x_p).$$

(26)

When $\tilde{w}^l$ is known, equation (26) is exactly in the same form as the following linear equation:

$$Y = A_0 + A_1 x_1 + A_2 x_2 + \cdots + A_p x_p$$

(27)

with $A_i$ in equation (27) corresponds $c_i^l \tilde{w}^l$ in equation (26). Equation (27) is exactly the fuzzy linear regression model expressing the linear relationship between the inputs, $x_i$ and the output $Y$ with $A_i$, $i = 1, 2, \ldots, p$, as the fuzzy parameters. This fuzzy linear regression model has been solved by Tanaka and coworkers [10,11,16] by the use of linear programming. Thus, we can use the same approach to identify the consequence parameters which are $c_i^l$, $i = 0, 1, \ldots, p$, $l = 1, 2, \ldots, m$. Following Tanaka’s approach except using the parameters in equation (26), the following linear programming problem can be formulated:

$$\min \sum_{k=1}^{N} \sum_{l=1}^{m} \sum_{i=0}^{p} \tilde{w}^l b_i^l x_{ik}$$

s.t. $b_i^l \geq 0$, $i = 0, \ldots, p$, $l = 1, \ldots, m$;

$$\sum_{i=0}^{p} \sum_{l=1}^{m} \tilde{w}^l a_i^l x_{ik} + (1 - \alpha) \sum_{i=1}^{m} \sum_{l=0}^{p} \tilde{w}^l b_i^l x_{ik} \geq y_k + (1 - \alpha) e_k,$$

$$- \sum_{i=0}^{p} \sum_{l=1}^{m} \tilde{w}^l a_i^l x_{ik} + (1 - \alpha) \sum_{i=1}^{m} \sum_{l=0}^{p} \tilde{w}^l b_i^l x_{ik} \geq -y_k + (1 - \alpha) e_k,$$

$k = 1, \ldots, N.$

(28)

The constraints in the above equation are used to satisfy the following inclusion condition:

$$[Y_k]_{\alpha} \subseteq [\tilde{Y}_k]_{\alpha}, \quad k = 1, \ldots, N,$$

(29)
where $\hat{Y}_k$ is the estimate of the $k^{th}$ observation $Y_k$, and $[.]_\alpha$ is the $\alpha$-level set. The above condition means that all the $\alpha$-level set of the given samples should be included in the $\alpha$-level set of the fuzzy model. Thus, by using the Tanaka approach, we also obtained an added advantage, namely, the results automatically satisfy the inclusion condition used in most possibilistic linear regression formulations.

4.2.2. The premise parameters

The premise parameters were trained by the use of the back propagation algorithm. The error measures at Layer 5 is back propagation to Layer 1 and the premise parameters are updated by a gradient descent method according to the error of back propagation. Since the purpose of training the premise parameters is to adjust the position and shape of the associated membership function in Layer 1 so that the density of the input functions can be represented, the spread of the membership function is not the concern of this training and thus only the first part of the error function $E$, that is, only the mode or the center and ignoring the influences of the spread, is used for calculating the back propagation error

$$E = \frac{1}{N} \sum_{k=1}^{N} e_k^2 = \frac{1}{N} \sum_{k=1}^{N} (y_k - \hat{y}_k)^2. \quad (30)$$

Using equation (30), the back propagation error for each layer can be calculated as follows.

Consider the $k^{th}$ error, that is, the error between the $k^{th}$ estimated and desired outputs, the error signal for the final output node can be calculated directly as

$$\epsilon_{5,1} = \frac{\partial e_k^2}{\partial y_k} = -2(y_k - \hat{y}_k), \quad (31)$$

where $\epsilon_{r,l}$ denotes the back propagation error of $l^{th}$ node in $r^{th}$ layer and thus $\epsilon_{5,1}$ denotes the error signal for the final output node. According to Jang and Sun [17], the back propagation error of each node in each layer is

$$\epsilon_{r,l} = \sum_{h=1}^{M_{r+1}} \epsilon_{r+1,h} \frac{\partial F_{r+1,h}}{\partial f_{r,l}}, \quad (32)$$

where $F_{r+1,h}$ represents the node function at the $h^{th}$ node of the $(r+1)^{th}$ layer, $f_{r,l}$ is the output of $l^{th}$ node of the $r^{th}$ layer, and $M_{r+1}$ is the total number of nodes in the $(r+1)^{th}$ layer.

The gradient vector is defined as the derivative of the error measure with respect to each parameter. If $\rho$ is a parameter of the $l^{th}$ node at layer $r$, we have

$$\frac{\partial e_k^2}{\partial \rho} = \epsilon_{r,l} \frac{\partial f_{r,l}}{\partial \rho}. \quad (33)$$

The derivative of the overall error measure $E$ with respect to $\rho$ is

$$\frac{\partial E}{\partial \rho} = \frac{1}{N} \sum_{k=1}^{N} \frac{\partial e_k^2}{\partial \rho}. \quad (34)$$

Thus, the updating formula for $\rho$ is

$$\Delta \rho = -\eta \frac{\partial E}{\partial \rho}, \quad (35)$$

where $\eta$ is the learning rate.

More details are given in the Appendix where the formulation of the learning rules for the premise parameters are illustrated by an example.
From the above discussion, the learning algorithm for FAN can be summarized as follows.

Step 0. Initialization
Set $\alpha$ value;
Determine the initial values of the premise parameter set $\{v_{i,j}, \sigma_{i,j}\}$ subjectively.

Step 1. Identify the consequent parameter set $\{c_i\}$ by solving the LP problem (28).

Step 2. Calculate the error measure $E$ according to (25). If $E$ is less than a pre-specified small allowed error, then stop; otherwise go to Step 3.

Step 3. Calculate the back propagation error by equations (31) and (32).

Step 4. Go to Step 1.

5. NUMERICAL EXAMPLES

To illustrate the approach, the following three artificial functions are used to generate the training data:

$$f_1(x_1, x_2) = 24.234r^2 (0.75 - r^2) + 5,$$  \hfill (36)

where $r^2 = (x_1/10 - 0.5)^2 + (x_2/10 - 0.5)^2$.

$$f_2(x_1, x_2) = 1.3356 \left(1.5 \left(1 - \frac{x_1}{10}\right) + e^{x_1/5} \sin \left(3\pi \left(\frac{x_1}{10} - 0.6\right)^2\right) + e^{3(x_2/10-0.5)} \sin \left(4\pi \left(\frac{x_2}{10} - 0.9\right)^2\right)\right) + 4,$$  \hfill (37)

and

$$f_3(x_1, x_2) = 10 \cdot \frac{\sin(2(x_1 - 5))}{2(x_1 - 5)} \cdot \frac{\sin(2(x_2 - 5))}{2(x_2 - 5)} + 5.$$  \hfill (38)

$f_1$ and $f_2$ are adopted and modified from Hwang et al. [18] and $f_3$ is adopted and modified from Jang [12]. Thirty pairs of input-output data are generated for each function. For simplicity, these data will be identified as Samples 1–3 for functions $f_1$, $f_2$, and $f_3$, respectively. Each set of data is generated in the following manner.

The input $(x_{k1}, x_{k2})$, $k = 1, \ldots, 30$, are generated as uniform random variates on $[0, 10]^2$. The mode of the fuzzy target output $Y_k$ is generated by

$$y_k = f_j(x_{k1}, x_{k2}), \quad \text{for } k = 1, \ldots, 30, \quad j = 1, 2, 3.$$  \hfill (39)

The corresponding spread of the target is

$$e_k = \frac{1}{4} y_k, \quad \text{for } k = 1, \ldots, 30.$$  \hfill (40)

Based on these data and the three functions, the fuzzy inference system is formulated as follows:

$$R^1 : \text{if } (x_1 \text{ is small}_1 \text{ and } x_2 \text{ is small}_2), \quad \text{then } (Y = Y^1 = c_1^1 + c_1^3 x_1 + c_1^4 x_2),$$

$$R^2 : \text{if } (x_1 \text{ is small}_1 \text{ and } x_2 \text{ is medium}_2), \quad \text{then } (Y = Y^2 = c_2^2 + c_2^3 x_1 + c_2^4 x_2),$$

$$R^3 : \text{if } (x_1 \text{ is small}_1 \text{ and } x_2 \text{ is large}_2), \quad \text{then } (Y = Y^3 = c_3^3 + c_3^1 x_1 + c_3^2 x_2),$$

$$R^4 : \text{if } (x_1 \text{ is medium}_1 \text{ and } x_2 \text{ is small}_2), \quad \text{then } (Y = Y^4 = c_4^4 + c_4^1 x_1 + c_4^2 x_2),$$

$$R^5 : \text{if } (x_1 \text{ is medium}_1 \text{ and } x_2 \text{ is medium}_2), \quad \text{then } (Y = Y^5 = c_5^5 + c_5^3 x_1 + c_5^4 x_2),$$

$$R^6 : \text{if } (x_1 \text{ is medium}_1 \text{ and } x_2 \text{ is large}_2), \quad \text{then } (Y = Y^6 = c_6^6 + c_6^2 x_1 + c_6^3 x_2),$$

$$R^7 : \text{if } (x_1 \text{ is large}_1 \text{ and } x_2 \text{ is small}_2), \quad \text{then } (Y = Y^7 = c_7^7 + c_7^1 x_1 + c_7^2 x_2),$$

$$R^8 : \text{if } (x_1 \text{ is large}_1 \text{ and } x_2 \text{ is medium}_2), \quad \text{then } (Y = Y^8 = c_8^8 + c_8^2 x_1 + c_8^3 x_2),$$

$$R^9 : \text{if } (x_1 \text{ is large}_1 \text{ and } x_2 \text{ is large}_2), \quad \text{then } (Y = Y^9 = c_9^9 + c_9^1 x_1 + c_9^2 x_2),$$

where each of the two input variables are partitioned into three partitions.
Table 1. Assumed initial values of the premise parameters.

<table>
<thead>
<tr>
<th></th>
<th>Sample 1</th>
<th>Sample 2</th>
<th>Sample 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\nu$</td>
<td>$\sigma$</td>
<td>$\nu$</td>
<td>$\sigma$</td>
</tr>
<tr>
<td>Small1</td>
<td>1</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>Medium1</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Large1</td>
<td>9</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>Small2</td>
<td>1</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>Medium2</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Large2</td>
<td>6</td>
<td>5</td>
<td>8</td>
</tr>
</tbody>
</table>

Figure 2. Convergence behavior with Sample 1.

This inference system is represented by a FAN network with two input nodes $x_1$ and $x_2$ and five layers. The nodes in Layer 1 are divided into two subgroups with three nodes each corresponding to the two input variables $x_1$ and $x_2$. Layers 2–4 all have nine nodes. The functions of the nodes in Layers 1 and 4 represent the parameters in the premise and the consequence sections, respectively. The nodes in other layers are fixed nodes. Finally, Layer 5 sums all the outputs from Layer 4 with a single output node.

The node function in Layer 1 are represented by the Gaussian membership function with parameters $v$ and $\sigma$, where $v$ represents the center and $\sigma$ represents the width. The assumed initial values for these centers and widths before training for the three sets of data are listed in Table 1.

Using Samples 1–3, the network is trained with a computer program run on the SUN IPX workstation. The convergence behaviors of the training of this network for the three different sets of data are plotted in Figures 2–4, where the "error" is defined as in equation (25) and an "epoch" means a complete presentation of the entire set of the training data.

From Figures 2–4, we can see that the performance for the three sets of data is consistent. The error measures for the three sets of data are all dropped fairly fast during the earlier iterations and then the convergence rate slowed down. The target outputs and the network or estimated outputs for the three different sets of data after training are listed in Tables 2–4. It can be seen
that the estimated outputs are very near the target outputs. In fact, some of the target outputs are reproduced by the estimated outputs. Furthermore, we can see from Tables 2-4 that the inclusion condition (29) is completely satisfied by the estimated outputs, that is, the $\alpha$-cut of the network outputs always cover the $\alpha$-cut of the target outputs.

The values of the premise parameters for the three sets of data after training are listed Table 5.

The convergence rates, the error measures and the computational time with the three sets of data are listed in Table 6, where $E$ denotes the error measure as defined in equation (25), CPU time is the time consuming by the computer for performing these training tasks and it is counted in seconds, and APE represents the average percentage error which is defined as

$$
APE = \frac{1}{N} \sum_{k=1}^{N} \left( \frac{y_k^{U} - \bar{y}_k^{U} + \bar{y}_k^{L} - y_k^{L}}{y_k^{U} - y_k^{L}} \right) \cdot 100%.
$$
where $y^U_k$ and $y^L_k$ are the upper and lower limits of $[Y_k]_\alpha$, $\bar{y}^U_k$ and $\bar{y}^L_k$ are the upper and lower limits of $[\bar{Y}_k]_\alpha$, and

$$
y^U_k = y_k + (1 - \alpha) e_k, \quad y^L_k = y_k - (1 - \alpha) e_k,
\bar{y}^U_k = \bar{y}_k + (1 - \alpha) \bar{e}_k, \quad \bar{y}^L_k = \bar{y}_k - (1 - \alpha) \bar{e}_k.
$$

The values of the APE of some of the illustrated examples are fairly large. The main reason for this large error is due to the use of the inclusion condition, equation (29), which causes a large spread in the results. If we only consider the modes, that is, defining the average percentage error as

$$
\text{APE}' = \frac{1}{N} \sum_{k=1}^{N} \frac{|y_k - \bar{y}_k|}{y_k},
$$

then the average percentage errors are reduced to 0.3%, 2%, and 6% for Samples 1–3, respectively.

Table 2. Results after training with Sample 1.

<table>
<thead>
<tr>
<th>Input Values</th>
<th>Target Outputs</th>
<th>Network Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$x_2$</td>
<td>$y$</td>
</tr>
<tr>
<td>3.178</td>
<td>7.145</td>
<td>7.734</td>
</tr>
<tr>
<td>3.553</td>
<td>4.261</td>
<td>7.481</td>
</tr>
<tr>
<td>1.194</td>
<td>6.210</td>
<td>3.926</td>
</tr>
<tr>
<td>2.639</td>
<td>5.727</td>
<td>7.278</td>
</tr>
<tr>
<td>9.736</td>
<td>8.843</td>
<td>17.832</td>
</tr>
<tr>
<td>3.711</td>
<td>3.831</td>
<td>6.900</td>
</tr>
<tr>
<td>2.830</td>
<td>3.631</td>
<td>5.800</td>
</tr>
<tr>
<td>0.891</td>
<td>7.150</td>
<td>3.714</td>
</tr>
<tr>
<td>6.349</td>
<td>2.660</td>
<td>10.115</td>
</tr>
<tr>
<td>7.188</td>
<td>1.798</td>
<td>11.475</td>
</tr>
<tr>
<td>1.300</td>
<td>4.852</td>
<td>4.117</td>
</tr>
<tr>
<td>0.516</td>
<td>5.976</td>
<td>3.249</td>
</tr>
<tr>
<td>0.725</td>
<td>2.129</td>
<td>4.132</td>
</tr>
<tr>
<td>1.071</td>
<td>7.482</td>
<td>4.463</td>
</tr>
<tr>
<td>8.931</td>
<td>7.540</td>
<td>15.267</td>
</tr>
<tr>
<td>7.106</td>
<td>2.084</td>
<td>11.245</td>
</tr>
<tr>
<td>2.967</td>
<td>8.528</td>
<td>6.784</td>
</tr>
<tr>
<td>3.609</td>
<td>8.893</td>
<td>8.979</td>
</tr>
<tr>
<td>3.161</td>
<td>7.127</td>
<td>8.085</td>
</tr>
<tr>
<td>3.161</td>
<td>7.127</td>
<td>8.085</td>
</tr>
<tr>
<td>5.802</td>
<td>0.217</td>
<td>8.913</td>
</tr>
<tr>
<td>4.304</td>
<td>3.582</td>
<td>6.910</td>
</tr>
<tr>
<td>0.807</td>
<td>9.837</td>
<td>4.860</td>
</tr>
</tbody>
</table>

$y$: mode to target output. $e$: spread of target output. $ar{y}$: mode to network output. $\bar{e}$: spread of network output.
Table 3. Results after training with Sample 2.

<table>
<thead>
<tr>
<th>Input Values</th>
<th>Target Outputs</th>
<th>Network Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$x_2$</td>
<td>$y$</td>
</tr>
<tr>
<td>8.486</td>
<td>0.674</td>
<td>8.301</td>
</tr>
<tr>
<td>3.613</td>
<td>9.075</td>
<td>7.535</td>
</tr>
<tr>
<td>4.104</td>
<td>4.537</td>
<td>5.182</td>
</tr>
<tr>
<td>0.379</td>
<td>2.430</td>
<td>8.187</td>
</tr>
<tr>
<td>8.839</td>
<td>5.784</td>
<td>7.219</td>
</tr>
<tr>
<td>5.998</td>
<td>1.443</td>
<td>7.029</td>
</tr>
<tr>
<td>7.971</td>
<td>4.760</td>
<td>6.423</td>
</tr>
<tr>
<td>1.081</td>
<td>4.159</td>
<td>7.295</td>
</tr>
<tr>
<td>8.395</td>
<td>2.577</td>
<td>7.429</td>
</tr>
<tr>
<td>2.009</td>
<td>1.523</td>
<td>7.751</td>
</tr>
<tr>
<td>8.278</td>
<td>5.040</td>
<td>6.674</td>
</tr>
<tr>
<td>0.473</td>
<td>5.337</td>
<td>7.717</td>
</tr>
<tr>
<td>5.365</td>
<td>1.823</td>
<td>6.605</td>
</tr>
<tr>
<td>6.386</td>
<td>4.522</td>
<td>5.379</td>
</tr>
<tr>
<td>8.106</td>
<td>3.923</td>
<td>6.681</td>
</tr>
<tr>
<td>8.345</td>
<td>1.463</td>
<td>7.946</td>
</tr>
<tr>
<td>3.206</td>
<td>9.349</td>
<td>7.835</td>
</tr>
<tr>
<td>4.795</td>
<td>1.650</td>
<td>6.740</td>
</tr>
<tr>
<td>2.017</td>
<td>3.649</td>
<td>6.670</td>
</tr>
<tr>
<td>9.915</td>
<td>4.733</td>
<td>7.981</td>
</tr>
<tr>
<td>2.672</td>
<td>3.086</td>
<td>6.451</td>
</tr>
<tr>
<td>9.035</td>
<td>8.424</td>
<td>8.190</td>
</tr>
<tr>
<td>3.285</td>
<td>1.113</td>
<td>7.491</td>
</tr>
<tr>
<td>5.397</td>
<td>2.195</td>
<td>6.302</td>
</tr>
<tr>
<td>3.082</td>
<td>4.163</td>
<td>5.749</td>
</tr>
<tr>
<td>0.953</td>
<td>8.301</td>
<td>8.155</td>
</tr>
<tr>
<td>1.747</td>
<td>0.926</td>
<td>8.150</td>
</tr>
<tr>
<td>6.550</td>
<td>5.012</td>
<td>5.422</td>
</tr>
<tr>
<td>8.583</td>
<td>9.516</td>
<td>8.364</td>
</tr>
<tr>
<td>5.200</td>
<td>5.181</td>
<td>5.013</td>
</tr>
</tbody>
</table>

$y$: mode to target output. $e$: spread of target output. $\hat{y}$: mode to network output. $\hat{e}$: spread of network output.

6. COMPARISON BETWEEN FAN AND FRBFN

The proposed fuzzy adaptive network FAN is closely related to the fuzzy radial basis function network FRBFN [8]. It has been pointed out by Jang and Sun [9] that the crisp radial basis function network and the Sugeno inference system are functional equivalent under certain conditions. The conditions to support this functional equivalence are as follows.

1. The number of hidden units in FRBFN is equal to the number of the fuzzy if-then rules or the number of nodes in Layer 4.
2. The consequence part of the fuzzy if-then rule is zero order. In another words, it must be a constant function not a linear function.
3. The radial function associated with each hidden unit of FRBFN has the same form of membership function associated with the linguistic variables in the fuzzy if-then rule.
4. The fuzzy and operator used to compute each rule’s firing strength must be multiplication.
Table 4. Results after training with Sample 3.

<table>
<thead>
<tr>
<th>Input Values</th>
<th>Target Outputs</th>
<th>Network Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$x_2$</td>
<td>$y$</td>
</tr>
<tr>
<td>6.859</td>
<td>9.688</td>
<td>4.992</td>
</tr>
<tr>
<td>7.215</td>
<td>0.617</td>
<td>4.849</td>
</tr>
<tr>
<td>3.199</td>
<td>2.898</td>
<td>5.256</td>
</tr>
<tr>
<td>0.260</td>
<td>9.151</td>
<td>4.994</td>
</tr>
<tr>
<td>5.528</td>
<td>7.114</td>
<td>3.275</td>
</tr>
<tr>
<td>3.539</td>
<td>2.766</td>
<td>4.837</td>
</tr>
<tr>
<td>9.476</td>
<td>8.443</td>
<td>5.042</td>
</tr>
<tr>
<td>5.968</td>
<td>9.446</td>
<td>5.276</td>
</tr>
<tr>
<td>7.562</td>
<td>2.678</td>
<td>5.384</td>
</tr>
<tr>
<td>5.103</td>
<td>3.324</td>
<td>4.379</td>
</tr>
<tr>
<td>2.802</td>
<td>6.101</td>
<td>4.208</td>
</tr>
<tr>
<td>2.831</td>
<td>6.492</td>
<td>4.887</td>
</tr>
<tr>
<td>8.287</td>
<td>6.081</td>
<td>5.167</td>
</tr>
<tr>
<td>5.479</td>
<td>2.999</td>
<td>3.382</td>
</tr>
<tr>
<td>0.423</td>
<td>0.885</td>
<td>5.033</td>
</tr>
<tr>
<td>8.976</td>
<td>1.165</td>
<td>5.160</td>
</tr>
<tr>
<td>2.316</td>
<td>7.121</td>
<td>5.310</td>
</tr>
<tr>
<td>0.080</td>
<td>2.127</td>
<td>5.036</td>
</tr>
<tr>
<td>2.937</td>
<td>1.300</td>
<td>4.755</td>
</tr>
<tr>
<td>5.408</td>
<td>1.343</td>
<td>6.047</td>
</tr>
<tr>
<td>6.512</td>
<td>6.041</td>
<td>5.163</td>
</tr>
<tr>
<td>3.504</td>
<td>5.534</td>
<td>5.409</td>
</tr>
<tr>
<td>9.319</td>
<td>1.516</td>
<td>5.075</td>
</tr>
<tr>
<td>6.879</td>
<td>2.906</td>
<td>5.318</td>
</tr>
<tr>
<td>6.793</td>
<td>0.142</td>
<td>5.035</td>
</tr>
<tr>
<td>8.325</td>
<td>2.432</td>
<td>4.904</td>
</tr>
<tr>
<td>0.539</td>
<td>8.211</td>
<td>5.012</td>
</tr>
<tr>
<td>1.544</td>
<td>6.900</td>
<td>4.863</td>
</tr>
<tr>
<td>9.298</td>
<td>2.566</td>
<td>4.826</td>
</tr>
</tbody>
</table>

$y$: mode to target output. $e$: spread of target output.
$\hat{y}$: mode to network output. $\hat{e}$: spread of network output.

Table 5. Values of the premise parameters after training.

<table>
<thead>
<tr>
<th>Sample 1</th>
<th>Sample 2</th>
<th>Sample 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\nu$</td>
<td>$\sigma$</td>
<td>$\nu$</td>
</tr>
<tr>
<td>Small</td>
<td>1.001</td>
<td>5.001</td>
</tr>
<tr>
<td>Medium</td>
<td>4.994</td>
<td>4.998</td>
</tr>
<tr>
<td>Large</td>
<td>9.002</td>
<td>4.995</td>
</tr>
<tr>
<td>Small</td>
<td>1.005</td>
<td>5.004</td>
</tr>
<tr>
<td>Medium</td>
<td>5.003</td>
<td>4.998</td>
</tr>
<tr>
<td>Large</td>
<td>6.003</td>
<td>4.998</td>
</tr>
</tbody>
</table>

Based on the above equivalence conditions, we can conclude that the capability of function approximation of FAN is superior to that of FRBFN, and the convergence rate of FAN is faster.
Table 6. Computational results for the training of FAN.

<table>
<thead>
<tr>
<th></th>
<th>Sample 1</th>
<th>Sample 2</th>
<th>Sample 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of epoch to converge</td>
<td>500</td>
<td>159</td>
<td>174</td>
</tr>
<tr>
<td>E</td>
<td>0.0148</td>
<td>0.0992</td>
<td>0.3469</td>
</tr>
<tr>
<td>APE</td>
<td>4%</td>
<td>15%</td>
<td>32%</td>
</tr>
<tr>
<td>CPU Time (sec)</td>
<td>297</td>
<td>70</td>
<td>85</td>
</tr>
</tbody>
</table>

E: error measure.
APE: Average Percentage Error.

Table 7. Computational results for the training of FRBFN.

<table>
<thead>
<tr>
<th></th>
<th>Sample 1</th>
<th>Sample 2</th>
<th>Sample 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of epoch to converge</td>
<td>358</td>
<td>368</td>
<td>496</td>
</tr>
<tr>
<td>E</td>
<td>0.05</td>
<td>0.88</td>
<td>0.87</td>
</tr>
<tr>
<td>APE</td>
<td>9%</td>
<td>45%</td>
<td>55%</td>
</tr>
<tr>
<td>CPU Time (sec)</td>
<td>57</td>
<td>67</td>
<td>91</td>
</tr>
</tbody>
</table>

E: error measure.
APE: Average Percentage Error.

than that of FRBFN. FAN is considered to have a better function approximation capability because of its linear form in the output of each rule, which of course will provide better nonlinear mapping than a constant form output. The tuning of the membership functions in FAN is always a one-dimensional search of the centers, whereas for FRBFN the search of the hidden unit centers is multi-dimensional when the input is multi-dimensional.

To test the above conclusions, the same data used previously is used to train the FRBFN network with ten hidden nodes. The results are listed in Table 7. Comparing the results between Tables 6 and 7, we can see that the accuracy and convergence rate of FAN are better than those of FRBFN. However, in general, the required computation time of FAN is greater than that of FRBFN. The reason for this is that FAN usually resulted in a larger sized linear programming problem in the learning of the consequence parameters.

7. CONCLUSIONS

One advantage of the proposed approach is that it satisfies the inclusion condition, equation (29), automatically. This condition can only be satisfied approximately for the back propagation neural network developed by Ishibuchi and Tanaka [4]. Furthermore, the fuzzy adaptive network is superior in both function approximation and convergence rate than those obtained by using the fuzzy radial basis function network. However, FAN needs to solve a larger linear programming problem than FRBFN. This is because that the FAN has a linear form output from each rule, while FRBFN only has zero order output from each hidden node. For example, if the FRBFN have m hidden nodes, then it needs to solve a linear programming problem with 2m variables at each iteration during the training. On the other hand, if the FAN have m rules (or m nodes in Layer 4) then it needs to solve a linear programming problem with 2(1 + p)m variables at each iteration during training, where p is the dimension of the input variables.
APPENDIX

Using the example in the fuzzy adaptive network section and let $F_1$, $F_2$, $F_3$, and $F_4$ represent the fuzzy terms “small”, “large”, “low”, and “high”, respectively. Define the membership functions as

$$
\mu_{F_1}(x_1) = \exp \left[ - \left( \frac{x_1 - v_1}{\sigma_1} \right)^2 \right],
$$

$$
\mu_{F_2}(x_1) = \exp \left[ - \left( \frac{x_1 - v_2}{\sigma_2} \right)^2 \right],
$$

$$
\mu_{F_3}(x_2) = \exp \left[ - \left( \frac{x_2 - v_3}{\sigma_3} \right)^2 \right],
$$

$$
\mu_{F_4}(x_2) = \exp \left[ - \left( \frac{x_2 - v_4}{\sigma_4} \right)^2 \right].
$$

Let the error measure for the $k$th input-output pair at the final output node as $(y_k - \hat{y}_k)^2$, then the back propagation error contributed by the $k$th pair at the final output node is

$$
\epsilon_{5,1} = \frac{\partial (y_k - \hat{y}_k)^2}{\partial \hat{y}_k} = -2(y_k - \hat{y}_k).
$$

The back propagation error of the $k$th pair for nodes in Layers 4, 3, 2, and 1 are determined according to equation (32) as follows.

**LAYER 4.**

$$
\epsilon_{4,l} = \epsilon_{5,1} \frac{\partial \sum_{h=1}^{4} w_{h} y_{h}^k}{\partial y_{l}^k} = \epsilon_{5,1}, \quad l = 1, 2, 3, 4,
$$

where $y_{l}^k$ is the mode of the fuzzy consequence $Y^l$ of the $l$th node in Layer 4 for the $k$th pair.

**LAYER 3.**

$$
\epsilon_{3,l} = \epsilon_{4,l} \frac{\partial \sum_{h=1}^{4} y_{l}^h}{\partial w_{l}^k} = \epsilon_{4,l} y_{l}^k, \quad l = 1, 2, 3, 4.
$$

**LAYER 2.**

$$
\epsilon_{2,1} = \sum_{h=1}^{4} \epsilon_{3,h} \frac{\partial \left( \frac{\sum_{h=1}^{4} w^h}{\sum_{l=1}^{4} w^l} \right)}{\partial w^1} = \epsilon_{3,1} \left( \frac{\sum_{l=1}^{4} w^l}{\sum_{l=1}^{4} w^l} \right)^2 - \epsilon_{3,2} \left( \frac{\sum_{l=1}^{4} w^l}{\sum_{l=1}^{4} w^l} \right) - \epsilon_{3,3} \left( \frac{\sum_{l=1}^{4} w^l}{\sum_{l=1}^{4} w^l} \right)^2 + \epsilon_{3,4} \left( \frac{\sum_{l=1}^{4} w^l}{\sum_{l=1}^{4} w^l} \right)^2.
$$

Similarly,

$$
\epsilon_{2,2} = \epsilon_{3,1} \left( \frac{\sum_{l=1}^{4} w^l}{\sum_{l=1}^{4} w^l} \right)^2 + \epsilon_{3,2} \left( \frac{\sum_{l=1}^{4} w^l}{\sum_{l=1}^{4} w^l} \right) + \epsilon_{3,3} \left( \frac{\sum_{l=1}^{4} w^l}{\sum_{l=1}^{4} w^l} \right)^2 + \epsilon_{3,4} \left( \frac{\sum_{l=1}^{4} w^l}{\sum_{l=1}^{4} w^l} \right)^2,
$$

$$
\epsilon_{2,3} = \epsilon_{3,1} \left( \frac{\sum_{l=1}^{4} w^l}{\sum_{l=1}^{4} w^l} \right)^2 + \epsilon_{3,2} \left( \frac{\sum_{l=1}^{4} w^l}{\sum_{l=1}^{4} w^l} \right) + \epsilon_{3,3} \left( \frac{\sum_{l=1}^{4} w^l}{\sum_{l=1}^{4} w^l} \right)^2 + \epsilon_{3,4} \left( \frac{\sum_{l=1}^{4} w^l}{\sum_{l=1}^{4} w^l} \right)^2.
$$
\[
\epsilon_{2,4} = \epsilon_{3,1} \frac{-w^1}{\left( \sum_{i=1}^{4} w^i \right)^2} + \epsilon_{3,2} \frac{-w^2}{\left( \sum_{i=1}^{4} w^i \right)^2} + \epsilon_{3,3} \frac{-w^3}{\left( \sum_{i=1}^{4} w^i \right)^2} + \epsilon_{3,4} \frac{-w^4}{\left( \sum_{i=1}^{4} w^i \right)^2}.
\]

**Layer 1.**

\[
\epsilon_{1,1} = \epsilon_{2,1} \left( \frac{\partial \mu_{F_1}(x_1) \times \mu_{F_3}(x_2)}{\partial \mu_{F_1}(x_1)} \right) + \epsilon_{2,2} \left( \frac{\partial \mu_{F_1}(x_1) \times \mu_{F_4}(x_2)}{\partial \mu_{F_1}(x_1)} \right) = \epsilon_{2,1} \mu_{F_3}(x_2) + \epsilon_{2,2} \mu_{F_4}(x_2),
\]

\[
\epsilon_{1,2} = \epsilon_{2,3} \left( \frac{\partial \mu_{F_1}(x_1) \times \mu_{F_3}(x_2)}{\partial \mu_{F_3}(x_2)} \right) + \epsilon_{2,4} \left( \frac{\partial \mu_{F_1}(x_1) \times \mu_{F_4}(x_2)}{\partial \mu_{F_3}(x_2)} \right) = \epsilon_{2,3} \mu_{F_3}(x_2) + \epsilon_{2,4} \mu_{F_4}(x_2),
\]

\[
\epsilon_{1,3} = \epsilon_{2,1} \left( \frac{\partial \mu_{F_2}(x_1) \times \mu_{F_3}(x_2)}{\partial \mu_{F_2}(x_1)} \right) + \epsilon_{2,3} \left( \frac{\partial \mu_{F_2}(x_1) \times \mu_{F_3}(x_2)}{\partial \mu_{F_2}(x_1)} \right) = \epsilon_{2,1} \mu_{F_3}(x_1) + \epsilon_{2,3} \mu_{F_3}(x_1),
\]

\[
\epsilon_{1,4} = \epsilon_{2,2} \left( \frac{\partial \mu_{F_2}(x_1) \times \mu_{F_4}(x_2)}{\partial \mu_{F_4}(x_2)} \right) + \epsilon_{2,4} \left( \frac{\partial \mu_{F_2}(x_1) \times \mu_{F_4}(x_2)}{\partial \mu_{F_4}(x_2)} \right) = \epsilon_{2,2} \mu_{F_3}(x_1) + \epsilon_{2,4} \mu_{F_3}(x_1).
\]

The training of the parameters \(\{(v_j, \sigma_j)\}_{j=1,...,4}\) is then controlled by the back propagation error in Layer 1. The descent direction or gradient is calculated by equation (33), and the updating rule for each parameter is formulated as

\[
\Delta v_1 = \eta \frac{\partial (y_k - \hat{y}_k)^2}{\partial v_1} = \eta_{e_{1,1}} \frac{\partial \mu_{F_1}(x_1)}{\partial v_1} = \eta_{e_{1,1}} \frac{\partial \exp \left[ -\left( \frac{x_1 - v_1}{\sigma_1} \right)^2 \right]}{\partial v_1} = 2\eta_{e_{1,1}} \exp \left[ -\left( \frac{x_1 - v_1}{\sigma_1} \right)^2 \right] \cdot \frac{x_1 - v_1}{\sigma_1^2},
\]

where \(\eta\) is the learning rate. And,

\[
\Delta \sigma_1 = \eta \frac{\partial (y_k - \hat{y}_k)^2}{\partial \sigma_1} = \eta_{e_{1,1}} \frac{\partial \mu_{F_1}(x_1)}{\partial \sigma_1} = \eta_{e_{1,1}} \frac{\partial \exp \left[ -\left( \frac{x_1 - v_1}{\sigma_1} \right)^2 \right]}{\partial \sigma_1} = 2\eta_{e_{1,1}} \exp \left[ -\left( \frac{x_1 - v_1}{\sigma_1} \right)^2 \right] \cdot \frac{(x_1 - v_1)^2}{\sigma_1^3}.
\]

Similarly,

\[
\Delta v_2 = 2\eta_{e_{1,2}} \exp \left[ -\left( \frac{x_1 - v_2}{\sigma_2} \right)^2 \right] \cdot \frac{x_1 - v_2}{(\sigma_2)^2},
\]

\[
\Delta \sigma_2 = 2\eta_{e_{1,2}} \exp \left[ -\left( \frac{x_1 - v_2}{\sigma_2} \right)^2 \right] \cdot \frac{(x_1 - v_2)^2}{(\sigma_2)^3},
\]

\[
\Delta v_3 = 2\eta_{e_{1,3}} \exp \left[ -\left( \frac{x_2 - v_3}{\sigma_3} \right)^2 \right] \cdot \frac{x_2 - v_3}{(\sigma_3)^2},
\]

\[
\Delta \sigma_3 = 2\eta_{e_{1,3}} \exp \left[ -\left( \frac{x_2 - v_3}{\sigma_3} \right)^2 \right] \cdot \frac{(x_2 - v_3)^2}{(\sigma_3)^3},
\]

\[
\Delta v_4 = 2\eta_{e_{1,4}} \exp \left[ -\left( \frac{x_2 - v_4}{\sigma_4} \right)^2 \right] \cdot \frac{x_2 - v_4}{(\sigma_4)^2},
\]

\[
\Delta \sigma_4 = 2\eta_{e_{1,4}} \exp \left[ -\left( \frac{x_2 - v_4}{\sigma_4} \right)^2 \right] \cdot \frac{(x_2 - v_4)^2}{(\sigma_4)^3}.
\]

The above learning rule is obtained by considering only the back propagation error contributed by the \(k\)th pair. For batch learning, the back propagation error is accumulated for all entries as shown in equation (34).
REFERENCES


