

Fuzzy Hyperbolic Regression With Fuzzy Neural Networks

¹M. Otadi, ²M. Mosleh, ³S. Saidanlu, ³N.A. Aris

¹Department of Mathematics, Kermanshah Branch, Islamic Azad University, Kermanshah, Iran.

²Department of Mathematics, Firoozkooh Branch, Islamic Azad University, Firoozkooh, Iran.

³Department of Mathematics Faculty of Science, University Teknologi Malaysia, 81310 UTM Skudai, Johor, Malaysia.

Abstract: Recently, fuzzy linear regression and fuzzy polynomial regression is considered by Mosleh *et al.*, (2010; 2011). In this paper, a novel hybrid method based on fuzzy neural network for approximate fuzzy coefficients (parameters) of fuzzy hyperbolic regression models with fuzzy output and crisp inputs, is presented. Here a neural network is considered as a part of a large field called neural computing or soft computing. Moreover, in order to find the approximate parameters, a simple algorithm from the cost function of the fuzzy neural network is proposed. Finally, we illustrate our approach by some numerical examples.

Key words: Fuzzy neural networks; Fuzzy regression model; Fuzzy number; Feedforward neural network; Learning algorithm.

INTRODUCTION

The concept of fuzzy numbers and fuzzy arithmetic operations were first introduced by Zadeh (1975) Dubois and Prade (1978). We refer the reader to (Kaufmann, 1985) for more information on fuzzy numbers and fuzzy arithmetic.

Regression analysis is of the most popular methods of estimation. It is applied to evaluate the functional relationship between the dependent and independent variables. Fuzzy regression analysis is an extension of the classical regression analysis in which some elements of the model are represented by fuzzy numbers. Fuzzy regression methods have been successfully applied to various problems such as forecasting (Chang, 1997; Chen 1999; Tseng, 2002; Ghavidel, 2011) and engineering (Lai, 1994). Thus, it is very important to develop numerical procedures that can appropriately treat fuzzy regression models. Modarres *et al.*, (2005) proposed a mathematical programming model to estimate the parameters of a fuzzy linear regression.

$$Y_i = A_1x_{i1} + A_2x_{i2} + A_nx_{in},$$

where $x_{ij} \in \mathbb{R}$ and $A_1, A_2, \dots, A_n, Y_i$ are symmetric fuzzy numbers for $i = 1, 2, \dots, m, j = 1, 2, \dots, n$.

Very recently, Mosleh *et al.*, (2010; 2011) proposed a learning algorithm of fuzzy neural network with crisp inputs, fuzzy weights and fuzzy output for adjusting fuzzy weights of fuzzy linear regression model and fuzzy polynomial regression model with fuzzy output and crisp inputs.

Ishibuchi *et al.*, (1995) proposed a learning algorithm of fuzzy neural networks with triangular fuzzy weights and Hayashi *et al.*, (1993) fuzzified the delta rule. Buckley and Eslami, (1997) consider neural net solutions to fuzzy problems. The topic of numerical solution of fuzzy polynomials by fuzzy neural network investigated by Abbasbandy *et al.*, (2006), consists of finding solution to polynomials like $a_1x + a_2x^2 + \dots + a_nx^n = a_0$ where $x \in \mathbb{R}$ and a_0, a_1, \dots, a_n are fuzzy numbers, and finding solution to systems of s fuzzy polynomial equations such as (Abbasbandy, 2008):

$$\begin{aligned} f_1(x_1, x_2, \dots, x_n) &= a_{10}, \\ &\vdots \\ f_i(x_1, x_2, \dots, x_n) &= a_{i0}, \\ &\vdots \\ f_s(x_1, x_2, \dots, x_n) &= a_{s0}, \end{aligned}$$

where $x_1, x_2, \dots, x_n \in \mathbb{R}$ and all coefficients are fuzzy numbers. Also, Otadi, *et al.*, (2011) proposed a learning algorithm of fuzzy neural network to estimate the solution of a fully fuzzy linear system.

In this paper, we first propose an architecture of fuzzy neural network (FNN) with fuzzy weights for real input vectors and fuzzy targets to find approximate coefficients to fuzzy hyperbolic regression model

$$\bar{Y}_i = \frac{1}{A_0 + A_1 x_{i1} + \dots + A_n x_{in}},$$

where i indexes the different observations, $x_{i1}, x_{i2}, \dots, x_{in} \in \mathbf{R}$, all coefficients and \bar{Y}_i are fuzzy numbers. The input-output relation of each unit is defined by the extension principle of Zadeh, (1975). Output from the fuzzy neural network, which is also a fuzzy number, is numerically calculated by interval arithmetic (Alefeld, 1983) for fuzzy weights and real inputs. Next, we define a cost function for the level sets of fuzzy outputs and fuzzy targets. Then, a crisp learning algorithm is derived from the cost function to find the fuzzy coefficients of the fuzzy hyperbolic regression model. The proposed algorithm is illustrated by some examples in the last section.

2 Preliminaries:

In this section the basic notations used in fuzzy calculus are introduced. We start by defining the fuzzy number.

Definition 1:

A fuzzy number is a fuzzy set $u: \mathbf{R}^1 \rightarrow I = [0,1]$ such that

- i. u is upper semi-continuous;
- ii. $u(x) = 0$ outside some interval $[a, d]$;
- iii. There are real numbers b and c , $a \leq b \leq c \leq d$, for which
 1. $u(x)$ is monotonically increasing on $[a, b]$,
 2. $u(x)$ is monotonically decreasing on $[c, d]$,
 3. $u(x) = 1, b \leq x \leq c$.

The set of all the fuzzy numbers (as given in definition 1) is denoted by E^1 .

An alternative definition which yields the same E^1 is given by (Kaleva 1987; Friedman, 1999).

Definition 2:

A fuzzy number u is a pair (\underline{u}, \bar{u}) of functions $\underline{u}(r)$ and $\bar{u}(r)$, $0 \leq r \leq 1$, which satisfy the following requirements:

- i. $\underline{u}(r)$ is a bounded monotonically increasing, left continuous function on $(0,1]$ and right continuous at 0;
- ii. $\bar{u}(r)$ is a bounded monotonically decreasing, left continuous function on $(0,1]$ and right continuous at 0;
- iii. $\underline{u}(r) \leq \bar{u}(r), 0 \leq r \leq 1$.

A crisp number r is simply represented by $\underline{u}(\alpha) = \bar{u}(\alpha) = r, 0 \leq \alpha \leq 1$. The set of all the fuzzy numbers is denoted by E^1 .

A popular fuzzy number is the triangular fuzzy number $u = (u_m, u_l, u_r)$ where u_m denotes the modal value and the real values $u_l > 0$ and $u_r > 0$ represent the left and right fuzziness, respectively. Its parametric form is $\underline{u}(\alpha) = u_m + u_l(\alpha - 1)$, $\bar{u}(\alpha) = u_m + u_r(1 - \alpha)$.

Triangular fuzzy numbers are fuzzy numbers in LR representation where the reference functions L and R are linear. The set of all triangular fuzzy numbers on R is called \hat{FZ} .

2.1 Operations on Fuzzy Numbers:

We briefly mention fuzzy number operations defined by the extension principle (Zadeh, 1975). Since coefficients vector of feedforward neural network is fuzzified in this paper, the operations we use in our fuzzy neural network are fuzzified by means of the extension principle. The h -level set of a fuzzy number X is defined by

$$[X]_h = \{x \in \mathbf{R} \mid \mu_X(x) \geq h\} \quad \text{for } 0 < h \leq 1,$$

and $[X]_0 = \bigcup_{h \in (0,1]} [X]_h$. Since level sets of fuzzy numbers become closed intervals, we denote $[X]_h$ by

$$[X]_h = [[X]_h^L, [X]_h^U],$$

where $[X]_h^L$ and $[X]_h^U$ are the lower and the upper limits of the h -level set $[X]_h$, respectively. The result of a fuzzy addition of triangular fuzzy numbers is a triangular fuzzy number again. So we only have to compute the following equation (Zimmermann, 1996):

$$(a_m, a_l, a_r) + (b_m, b_l, b_r) = (a_m + b_m, a_l + b_l, a_r + b_r),$$

also

$$\frac{(a_m, a_l, a_r)}{(b_m, b_l, b_r)} = \left(\frac{a_m}{b_m}, \frac{a_m}{b_m} - \frac{a_m - a_l}{b_m + b_r}, \frac{a_m}{b_m} + \frac{a_m + a_r}{b_m - b_l} \right).$$

From interval arithmetic (Alefeld, 1983), the above operations on fuzzy numbers are written for h -level sets as follows:

$$A = B \Leftrightarrow [A]_h = [B]_h \quad \text{for } 0 < h \leq 1, \quad (1)$$

$$[A + B]_h = [[A]_h^L + [B]_h^L, [A]_h^U + [B]_h^U], \quad (2)$$

$$[k.A]_h = [k.[A]_h^L, k.[A]_h^U], \quad k \geq 0$$

$$[k.A]_h = [k.[A]_h^U, k.[A]_h^L], \quad k < 0.$$

We describe the classical definition of distance between fuzzy numbers (Feuring, 1995).

Definition 3:

The mapping $\hat{d}: \hat{FZ} \times \hat{FZ} \rightarrow \mathbb{R}^+$ is defined by

$$\hat{d}(A, B) = \max(|a_m - b_m|, |a_l - b_l|, |a_r - b_r|),$$

where $A = (a_m, a_l, a_r)$ and $B = (b_m, b_l, b_r)$. It can be proved that \hat{d} is a metric on \hat{FZ} and so (\hat{FZ}, \hat{d}) becomes a metric space.

2.2 Input-Output Relation of Each Unit:

Let us fuzzify a two-layer feedforward neural network with n input units and one output unit. Input vectors, targets and connection weights are fuzzified (i.e., extended to fuzzy numbers). In order to derive a crisp learning rule, we restrict fuzzy weights, real inputs and fuzzy target within triangular fuzzy numbers.

The input-output relation of each unit of the fuzzified neural network can be written as follows.

Input units:

$$o_{i0} = 1, o_{ij} = x_{ij}, \quad j = 1, 2, \dots, n, i = 1, \dots, m. \quad (3)$$

Output unit:

$$\bar{Y}_i = f(Net_i), \quad i = 1, \dots, m, \quad (4)$$

$$Net_i = W_0 + o_{i1}W_1 + \dots + o_{in}W_n,$$

where x_{ij} is a real input and W_j is the fuzzy coefficient weight.

2.3 Calculation of Fuzzy Output:

The fuzzy output from each unit in Eqs.(3)-(4) is numerically calculated for level sets of fuzzy weights and real inputs. The input-output relations of our fuzzy neural network can be written for the h -level sets.

Input units:

$$o_{i0} = 1, o_{ij} = x_{ij}, \quad j = 1, 2, \dots, n, i = 1, \dots, m. \quad (5)$$

Output unit:

$$[\bar{Y}_i]_h = f([Net_i]_h), \quad i = 1, \dots, m, \quad (6)$$

$$[Net_i]_h = \sum_{j=0}^n o_{ij} \cdot [W_j]_h.$$

From Eqs.(5)-(6), we can see that the h -level sets of the fuzzy output \bar{Y}_i is calculated from those of the fuzzy inputs and fuzzy weights. The above relations are written as follows.

Input units:

$$o_{i0} = 1, o_{ij} = x_{ij}, \quad j = 1, 2, \dots, n, i = 1, \dots, m. \quad (7)$$

Output unit:

$$[\bar{Y}_i]_h = [[\bar{Y}_i]_h^L, [\bar{Y}_i]_h^U] = [f([Net_i]_h^L), f([Net_i]_h^U)], \quad (8)$$

where f is an increasing function.

$$[Net_i]_h = [[Net_i]_h^L, [Net_i]_h^U] =$$

$$[\sum_{j \in b} o_{ij} \cdot [W_j]_h^L + \sum_{j \in c} o_{ij} \cdot [W_j]_h^U, \sum_{j \in b} o_{ij} \cdot [W_j]_h^U + \sum_{j \in c} o_{ij} \cdot [W_j]_h^L], \quad i = 1, \dots, m, \quad (9)$$

where $b = \{j \mid o_{ij} \geq 0\}$, $c = \{j \mid o_{ij} < 0\}$ and $b \cup c = \{0, \dots, n\}$.

3 The Hyperbolic Regression Model:

We have postulated that the dependent fuzzy variable Y , is a function of the independent real variables x_1, x_2, \dots, x_n . More formally

$$f: \mathbb{R}^n \rightarrow E,$$

$$Y_i = f(x_{i1}, x_{i2}, \dots, x_{in}),$$

where i indexes the observations.

The objective is to estimate a fuzzy hyperbolic regression (FHR) model, express as follows:

$$Y_i = \frac{1}{A_0 + A_1 x_{i1} + A_2 x_{i2} + \dots + A_n x_{in}}. \quad (10)$$

Let A_0, A_1, \dots, A_n denote the list of regression coefficients (parameters) where A_1, \dots, A_n are weights or regression coefficients corresponding to x_{i1}, \dots, x_{in} . Then fuzzy hyperbolic regression is given by

$$\bar{Y}_i = \frac{1}{A_0 + A_1 x_{i1} + A_2 x_{i2} + \dots + A_n x_{in}}, \quad (11)$$

where i indexes the different observations and A_0, A_1, \dots, A_n are fuzzy numbers. We are interested in finding A_0, A_1, \dots, A_n of fuzzy hyperbolic regression such that $\frac{1}{Y_i}$ approximates $\frac{1}{Y_i}$ for all $i = 1, 2, \dots, m$, closely enough

according to some norm $\|\cdot\|$, i.e.,

$$\min \left\| \left[\frac{1}{Y_i} \right]_h^\dagger - \left[\frac{1}{Y_i} \right]_h^\dagger \right\|, \quad h \in [0, 1], \quad (12)$$

where \dagger means we have this equation for U (upper limit) and L (lower limit) together, independently. Also, we use this notation after this, anywhere. Therefore,

$$\min \hat{d}\left(\frac{1}{Y_i}, \frac{1}{Y_i}\right) \text{ for all } i = 1, 2, \dots, m. \quad (13)$$

Then, it becomes a problem of optimization.

A FNN_4 (fuzzy neural network with fuzzy weights, output signals and real inputs) solution to Eq. (11) is given in figure 1. The input neurons make no change in their inputs and the input signals interact with the weights, so the input to the output neuron is

$$A_0 + A_1 x_{i1} + \dots + A_n x_{in},$$

and the output, in the output neuron, equals its input, so

$$\frac{1}{Y_i} = A_0 + A_1 x_{i1} + \dots + A_n x_{in}.$$

How does the FNN_4 solve the fuzzy hyperbolic regression? The training data are $\{(1, x_{i1}, \dots, x_{in}), \dots, (1, x_{m1}, \dots, x_{mn})\}$ for inputs and target (desired) outputs are

. We proposed a learning algorithm from the cost function for adjusting fuzzy number weights.

Following Section 4, we proposed a learning algorithm such that the network can approximate the fuzzy A_0, A_1, \dots, A_n of Eq. (11) to any degree of accuracy.

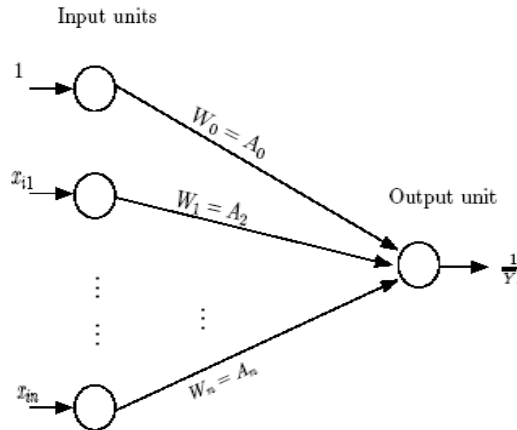


Fig. 1: Fuzzy neural network for approximating fuzzy hyperbolic regression.

3.1 Learning Fuzzy Neural Network:

Consider the learning algorithm of the two-layer fuzzy feedforward neural network with 2 inputs and one output as shown in figure 1. Let the h -level sets of the target output $Y_i, i = 1, \dots, m$ be denoted

$$[Y_i]_h = [[Y_i]_h^L, [Y_i]_h^U], \quad i = 1, \dots, m, \quad (14)$$

where $Y_i^L(h)$ shows the left-hand side and $Y_i^U(h)$ the right-hand side of the h -level sets of the desired output.

A cost function to be minimized is defined for each h -level sets as follows:

$$[E(W_0, W_1, \dots, W_n)]_h = [E(W_0, W_1, \dots, W_n)]_h^L + [E(W_0, W_1, \dots, W_n)]_h^U, \quad (15)$$

where

$$[E(W_0, W_1, \dots, W_n)]_h^\dagger = \frac{1}{2} \sum_{i=1}^m ([\frac{1}{Y_i}]_h^\dagger - [\frac{1}{Y_i}]_h^\dagger)^2.$$

The total cost function for the input-output pair $(x_i, \frac{1}{Y_i})$ is obtained as

$$e = \sum_h [E(W_0, W_1, \dots, W_n)]_h. \quad (16)$$

Hence $[E(W_0, W_1, \dots, W_n)]_h^L$ denotes the error between the left-hand sides of the h -level sets of the desired and the computed output, and $[E(W_0, W_1, \dots, W_n)]_h^U$ denotes the error between the right-hand sides of the h -level sets of the desired and the computed output.

In the research of neural networks, the norm is often defined as follows:

$$[E(W_0, W_1, \dots, W_n)]_h^\dagger = \frac{1}{2} \sum_{i=1}^m ([\frac{1}{Y_i}]_h^\dagger - [\frac{1}{Y_i}]_h^\dagger)^2 = \frac{1}{2} \sum_{i=1}^m (\sum_{j=0}^n [o_{ij} W_j]_h^\dagger - [\frac{1}{Y_i}]_h^\dagger)^2. \quad (17)$$

Clearly, this is a problem of optimization of quadratic functions without constraints that can usually be solved by gradient descent algorithm. In fact, denoting

$$[\nabla E(W)]_h^\dagger = ([\frac{\partial E(W)}{\partial W_0}]_h^\dagger, \dots, [\frac{\partial E(W)}{\partial W_n}]_h^\dagger)^T,$$

in order to solve Eq. (12), assume k iterations to have been done and get the k^{th} iteration point W_k .

REMARK 1. Since the Eq. (17) are quadratic functions, supposing $0 \leq o_{ij}$ for $i = 1, \dots, m$, $j = 0, \dots, n$, we rewrite these as follows:

$$\begin{aligned} [E(W)]_h^\dagger &= \frac{1}{2} \sum_{i=1}^m (\sum_{j=0}^n [o_{ij} W_j]_h^\dagger - [\frac{1}{Y_i}]_h^\dagger)^2 \\ &= \frac{1}{2} \sum_{i=1}^m [(\sum_{j=0}^n o_{ij} [W_j]_h^\dagger)^2 - 2[\frac{1}{Y_i}]_h^\dagger \sum_{j=0}^n o_{ij} [W_j]_h^\dagger + ([\frac{1}{Y_i}]_h^\dagger)^2] \\ &= \frac{1}{2} ([W]_h^\dagger)^T Q [W]_h^\dagger + ([B]_h^\dagger)^T [W]_h^\dagger + [C]_h^\dagger, \end{aligned}$$

where

$$Q = \begin{bmatrix} m & \sum_{i=1}^m o_{i1} & \sum_{i=1}^m o_{i2} & \dots & \sum_{i=1}^m o_{in} \\ \sum_{i=1}^m o_{i1} & \sum_{i=1}^m o_{i1}^2 & \sum_{i=1}^m o_{i1} o_{i2} & \dots & \sum_{i=1}^m o_{i1} o_{in} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \sum_{i=1}^m o_{in} & \sum_{i=1}^m o_{in} o_{i1} & \sum_{i=1}^m o_{in} o_{i2} & \dots & \sum_{i=1}^m o_{in}^2 \end{bmatrix}.$$

$$[B]_h^\dagger = ([b_0]_h^\dagger, [b_1]_h^\dagger, \dots, [b_n]_h^\dagger)^T,$$

$$[C]_h^\dagger = \frac{1}{2} \sum_{i=1}^m ([\frac{1}{Y_i}]_h^\dagger)^2,$$

with $[b_j]_h^\dagger = -\sum_{i=1}^m o_{ij} [\frac{1}{Y_i}]_h^\dagger$. We have

$$[\nabla E(W)]_h^L = Q[W]_h^L + [B]_h^L, \quad (18)$$

and

$$[\nabla E(W)]_h^U = Q[W]_h^U + [B]_h^U. \quad (19)$$

To find the stationary point of $[E(W)]_h = ([E(W)]_h^L, [E(W)]_h^U)$, we should put $[\nabla E(W)]_h^L = [\nabla E(W)]_h^U = (0, 0, \dots, 0)^T$. When Q is positive definite matrices, the stationary point can be obtained as follows:

$$[W^*]_h^\dagger = -Q^{-1}[B]_h^\dagger. \quad (20)$$

The Hessian matrices at this point are

$$[\nabla^2 E(W^*)]_h^\dagger = [\nabla(\nabla E(W^*))]_h^\dagger = \left[\begin{array}{cccc} \left[\frac{\partial^2 E(W)}{\partial W_0^2} \right]_h^\dagger & \left[\frac{\partial^2 E(W)}{\partial W_1 \partial W_0} \right]_h^\dagger & \cdots & \left[\frac{\partial^2 E(W)}{\partial W_n \partial W_0} \right]_h^\dagger \\ \vdots & \left[\frac{\partial^2 E(W)}{\partial W_1^2} \right]_h^\dagger & \cdots & \left[\frac{\partial^2 E(W)}{\partial W_n \partial W_1} \right]_h^\dagger \\ \cdots & \cdots & \cdots & \cdots \\ \vdots & \left[\frac{\partial^2 E(W)}{\partial W_1 \partial W_n} \right]_h^\dagger & \cdots & \left[\frac{\partial^2 E(W)}{\partial W_n^2} \right]_h^\dagger \end{array} \right]_{W=W^*} = Q,$$

which are positive definite matrices because Q is positive definite. From optimization theory, we know that $[W^*]_h = ([W^*]_h^L, [W^*]_h^U) = (-Q^{-1}[B]_h^L, -Q^{-1}[B]_h^U)$, is the unique solution of the problem.

REMARK 2. The above method is not very convenient in applications. Now we consider its explicit scheme. Since $[\nabla E(W)]_h^L = Q[W]_h^L + [B]_h^L$ and $[\nabla E(W)]_h^U = Q[W]_h^U + [B]_h^U$, then $[\nabla E(W_k)]_h^L = Q[W_k]_h^L + [B]_h^L$ and $[\nabla E(W_k)]_h^U = Q[W_k]_h^U + [B]_h^U$.

Hence we have (Ishibuchi, 1995; Ishibuchi, 2001; Rumelhart, 1986)

$$\begin{aligned} W_{k+1} &= W_k + \Delta W_k, \\ \Delta W_k &= -\mu \nabla E(W_k), \end{aligned} \quad (21)$$

where k indexes the number of adjustments and μ is a fuzzy learning rate (a positive real number).

We know that (Li, 2003).

$$([\nabla E(W_{k+1})]_h^\dagger)^T [\nabla E(W_k)]_h^\dagger = 0,$$

therefore we have (Ishibuchi, 1995)

$$(Q([W_k]_h^\dagger - [\mu_k]_h^\dagger [\nabla E(W_k)]_h^\dagger) + [B]_h^\dagger)^T (Q[W_k]_h^\dagger + [B]_h^\dagger) = 0.$$

Rearranging them, we have:

$$([\nabla E(W_k)]_h^\dagger - [\mu_k]_h^\dagger Q[\nabla E(W_k)]_h^\dagger)^T [\nabla E(W_k)]_h^\dagger = 0.$$

From these equations, we can easily get an expression for $[\mu_k]_h^L$ and $[\mu_k]_h^U$:

$$[\mu_k]_h^L = \frac{([\nabla E(W_k)]_h^L)^T [\nabla E(W_k)]_h^L}{([\nabla E(W_k)]_h^L)^T Q[\nabla E(W_k)]_h^L}, \quad (22)$$

and

$$[\mu_k]_h^U = \frac{([\nabla E(W_k)]_h^U)^T [\nabla E(W_k)]_h^U}{([\nabla E(W_k)]_h^U)^T Q[\nabla E(W_k)]_h^U}. \quad (23)$$

Substituting these into equations

$$W_{k+1} = W_k + \Delta W_k, \quad (24)$$

$$\Delta W_k = -\mu_k \nabla E(W_k),$$

where k indexes the number of adjustments and $\mu_k = ([\mu_k]_h^L, [\mu_k]_h^U)$ is a learning rate, we have the explicit scheme

$$[W_{k+1}]_h^L = [W_k]_h^L - \frac{([\nabla E(W_k)]_h^L)^T [\nabla E(W_k)]_h^L}{([\nabla E(W_k)]_h^L)^T Q [\nabla E(W_k)]_h^L} [\nabla E(W_k)]_h^L, \quad (25)$$

and

$$[W_{k+1}]_h^U = [W_k]_h^U - \frac{([\nabla E(W_k)]_h^U)^T [\nabla E(W_k)]_h^U}{([\nabla E(W_k)]_h^U)^T Q [\nabla E(W_k)]_h^U} [\nabla E(W_k)]_h^U. \quad (26)$$

We can also obtain similar relations for $o_{ij} < 0$ for $i = 1, \dots, m$, $j = 0, \dots, n$ and other cases.

4 Algorithm:

Step 1: Read K (number of iterations), W_j (fuzzy weights W_j are initialized values), $\{(x_{i1}, \dots, x_{in}, Y_i)\}$ for $i = 1, 2, \dots, m$, (real inputs and fuzzy target output).

Step 2: Compute $[E(W_0, \dots, W_n)]_h = [E(W_0, \dots, W_n)]_h^L + [E(W_1, \dots, W_n)]_h^U$.

Step 3: Read μ (real learning rate) or compute $\mu_k = ([\mu_k]_h^L, [\mu_k]_h^U)$ is a learning rate.

Step 4: The fuzzy weight $W^T = (W_0, \dots, W_n)^T$ is updated by the Eq. (21) or Eq. (24).

Step 5: If $k < K$ then $k := k + 1$ and we continue the training by going back to step 2, otherwise we go to step 6.

Step 6: The training cycle is completed.

5 Comparison With Other Methods:

This study would not be completed without comparing it with other existing methods. Some comparisons are as follows:

Mosleh *et al.*, (2010; 2011) have considered the fuzzy linear regression and fuzzy polynomial regression where input units are crisp numbers and output unit is a fuzzy number also Mosleh *et al.*, (2010) have considered the fully fuzzy linear regression, but in this paper we consider fuzzy hyperbolic regression

$$Y_i = \frac{1}{A_0 + A_1 x_{i1} + A_2 x_{i2} + \dots + A_n x_{in}}, \text{ where } x_{i1}, x_{i2}, \dots, x_{in} \text{ are real numbers and } Y_i \text{ is a fuzzy number.}$$

In this paper, if we consider fuzzy hyperbolic regression model with $Z_i = \frac{1}{Y_i}$, from the point of view of prediction, we have done this comparison between this paper and (Kao, 2003; Tanaka, 1989) in example 6.1.

6 Numerical Examples:

To illustrate the technique proposed in this paper, consider the following examples.

Example 6.1.

Kao *et al.*, and Tanaka *et al.*, (2003; 1989) used an example to illustrate their regression model, in that the explanatory variable is crisp and the responses are triangular fuzzy numbers. That example has five sets of the (x_i, Z_i) observations, see table 1. For each fuzzy numbers, we use 11 h-cuts $h = 0, 0.1, \dots, 1$, where we calculate the error of each fuzzy output by

$$e_{Z_i} = \frac{1}{2} \sum_h ([\bar{Z}_i]_h^L - [Z_i]_h^L)^2 + \frac{1}{2} \sum_h ([\bar{Z}_i]_h^U - [Z_i]_h^U)^2,$$

and total error by Eq. (16).

In the computer simulation of this example, we use the following specifications of the learning algorithm.

- (1) Number of input units: 2 units.
- (2) Number of output units: 1 unit.
- (3) Stopping condition: $K = 8$ iterations of the learning algorithm.

The training starts with $W_0(1) = (1, 0.5, 0.5)$ and $W_1(1) = (0.3, 0.3, 0.2)$. Applying the proposed method to the approximate solution of problem (11). In symbols, the fuzzy neural network model is:

$$\bar{Z}_{FNN} = (4.9499, 1.8399, 1.8398) + (1.71, 0.16, 0.1601)x_1.$$

In the study of Tanaka *et al.*, (1989) the results of the Min problem of $h = 0$ is used for comparison. The fuzzy regression model is:

$$\bar{Z}_T = (3.850, 3.850, 3.850) + 2.100x_1.$$

In the study of Kao *et al.*, (2003) the results of the fuzzy regression model is:

$$\bar{Z}_K = 4.808 + 1.718x_1 + (0.118, 2.32, 2.32).$$

To compare the performance of these three methods in estimation, we apply to calculate the errors in estimating the observed responses. Table 1 shows the errors in estimating the five observation for these three methods. The total error of the fuzzy neural network method is 79.0118, which is obviously better than the total error of 81.6567 calculated from the Kao method and the total error of 186.8226 calculated from the Tanaka method.

Table 1: Numerical data and the estimation errors for example 6.1.

Independent variable	Response variable	Errors in estimation		
		Tanaka	Kao	Neuralnetwork
1	(8.0,1.8,1.8)	62.4071	21.2671	19.9084
2	(6.4,2.2,2.2)	62.4071	21.2671	19.9084
3	(9.5,2.6,2.6)	10.6631	4.0022	4.0016
4	(13.5,2.6,2.6)	23.2031	32.1667	32.2214
5	(13.0,2.4,2.4)	28.1421	2.9535	2.9719
Total error		186.8226	81.6567	79.0118

Example 6.2:

Consider the fuzzy data for a dependent fuzzy variable Y and two independent real variables X_1 and X_2 in table 2.

Using these data, develop an estimated fuzzy regression equation $\bar{Y} = \frac{1}{A_0 + A_1x_1 + A_2x_2}$.

Table 2: Inputs and output data for example 6.2.

x_1	12	7	4	5	2
x_2	7	3	4	8	1
Y	(2,1,1)	(0.2,0.1,0.2)	(1,0.5,1)	(0.8,0.2,0.3)	(1.1,0.1,0.1)

In the computer simulation of this example, we use the following specifications of the learning algorithm.

- (1) Number of input units: 3 units.
- (2) Number of output units: 1 unit.
- (3) Stopping condition: $K = 15$ iterations of the learning algorithm.

The training starts with $W_0(1) = (1,0.5,0.5)$, $W_1(1) = (1,1,1)$ and $W_2(1) = (-0.2,0.1,0.1)$. Applying the proposed method to the approximate solution of problem (11). Table 3 show the convergence behavior in the 15 iterations. Therefor we have fuzzy regression equation

$$\bar{Y} = \frac{1}{(2.237, 0.871, 4.986) + (0.151, 0.112, 0.333)x_1 + (-0.307, 0.182, 0.634)x_2}.$$

Table 3: Numerical results for example 6.2.

k	$W_0(k)$	$W_1(k)$	$W_2(k)$
1	(1,0.5,0.5)	(1,1,1)	(-0.2,0.1,0.1)
2	(1.520,1.072,1.006)	(1.094,1.304,1.511)	(-0.194,0.567,0.830)
3	(1.505,1.529,1.304)	(1.5642,1.113,0.518)	(-0.610,0.586,1.599)
⋮	⋮	⋮	⋮
15	(2.237,0.871,4.986)	(0.151,0.112,0.333)	(-0.307,0.182,0.634)

Summary and Conclusions:

Solving fuzzy hyperbolic regression (FHR) by using universal approximators (UA), that is, FNN is presented in this paper. The problem formulation of the proposed UAM is quite straightforward. To obtain the "Best-approximated" solution of FHRs, the adjustable parameters of FNN are systematically adjusted by using the learning algorithm.

In this paper, we derived a learning algorithm of fuzzy weights of two-layer feedforward fuzzy neural networks whose input-output relations were defined by extension principle. The effectiveness of the derived learning algorithm was demonstrated by computer simulation of numerical examples. Computer simulation in this paper was performed for two-layer feedforward neural networks using the back-propagation-type learning algorithm.

REFERENCES

- Abbasbandy, S. and M. Otadi, 2006. Numerical solution of fuzzy polynomials by fuzzy neural network, *Appl. Math. Comput.*, 181: 1084-1089.
- Abbasbandy, S., M. Otadi and M. Mosleh, 2008. Numerical solution of a system of fuzzy polynomials by fuzzy neural network, *Inform. Sci.*, 178: 1948-1960.
- Alefeld, G. and J. Herzberger, 1983. *Introduction to Interval Computations*, Academic Press, New York, 1983.
- Buckley, J.J. and E. Eslami, 1997. Neural net solutions to fuzzy problems: The quadratic equation, *Fuzzy Sets Syst.*, 86: 289-298.
- Chang, P.T., 1997. Fuzzy seasonality forecasting, *Fuzzy Sets Syst.*, 90: 1-10.
- Chen, T. and M.J.J. Wang, 1999. Forecasting methods using fuzzy concepts, *Fuzzy Sets Syst.*, 105: 339-352.
- Dubois, D. and H. Prade, 1978. Operations on fuzzy numbers, *J. Systems Sci.*, 9: 613-626.
- Feuring T.H. and W.M. Lippe, 1995. Fuzzy neural networks are universal approximators, *IFSA World Congress 1995*, Sao Paulo, Brasil, 2: 659-662.
- Ghavidel, S., M. Otadi and M. Mosleh, 2011. Evaluation of fuzzy labor market by fuzzy neural network, *Australian Journal of Basic and Applied Sciences*, 5: 266-285.
- Hayashi, Y., J.J. Buckley and E. Czogala, 1993. Fuzzy neural network with fuzzy signals and weights, *Internat. J. Intelligent Systems*, 8: 527-537.
- Ishibuchi, H., K. Kwon and H. Tanaka, 1995. A learning algorithm of fuzzy neural networks with triangular fuzzy weights, *Fuzzy Sets Syst.*, 71: 277-293.
- Ishibuchi, H. and M. Nii, 2001. Numerical analysis of the learning of fuzzified neural networks from fuzzy if-then rules, *Fuzzy Sets Syst.*, 120: 281-307.
- Kaleva, O., 1987. Fuzzy differential equations, *Fuzzy Sets Syst.*, 24: 301-317.
- Kao, C. and C.L. Chyu, 2003. Least-squares estimates in fuzzy regression analysis, *European J. Oper. Res.* 148: 426-435.
- Kaufmann, A. and M.M. Gupta, 1985. *Introduction Fuzzy Arithmetic*, Van Nostrand Reinhold, New York, 1985.
- Lai, Y.J. and S.I. Chang, 1994. A fuzzy approach to multiperson optimization: an off-line quality engineering problem, *Fuzzy Sets Syst.*, 63: 117-129.
- Li, H.X., L.X. Li and J.Y. Wang, 2003. Interpolation functions of feedforward neural networks, *Comput. Math. Appl.*, 46: 1861-1874.
- Ma, M., M. Friedman and A. Kandel, 1999. A new fuzzy arithmetic, *Fuzzy Sets Syst.*, 108: 83-90.
- Modarres, M., E. Nasrabadi and M.M. Nasrabadi, 2005. Fuzzy linear regression models with least square errors, *Appl. Math. Comput.*, 163: 977-989.
- Mosleh, M., T. Allahviranloo and M. Otadi, Evaluation of fully fuzzy regression models by fuzzy neural network, *Neural Computing and Applications*, In press.
- Mosleh, M., M. Otadi, S. Abbasbandy, 2010. Evaluation of fuzzy regression models by fuzzy neural network, *J. Comput. Appl. Math.*, 234: 825-834.
- Mosleh, M., M. Otadi and S. Abbasbandy, 2011. Fuzzy polynomial regression with fuzzy neural networks, *Applied mathematical modelling*, 35: 5400-5412.
- Otadi, M. and M. Mosleh, 2011. Simulation and evaluation of dual fully fuzzy linear systems by fuzzy neural network, *Applied mathematical modelling*, 35: 5026-5039.
- Otadi, M., M. Mosleh and S. Abbasbandy, 2011. Numerical solution of fully fuzzy linear systems by fuzzy neural network, *Soft computing*, 15: 1513-1522.
- Rumelhart, D.E., J.L. McClelland, 1986. and the PDP Research Group, *Parallel Distributed Processing*, Vol. 1, MIT Press, Cambridge, MA.
- Tanaka, H., I. Hayashi and J. Watada, 1989. Possibilistic linear regression analysis for fuzzy data, *European J. Oper. Res.*, 40: 389-396.
- Tseng, F.M. and G.-H. Tzeng, 2002. A fuzzy seasonal ARIMA model for forecasting, *Fuzzy Sets Syst.*, 126: 367-376.
- Zadeh, L.A., 1975. The concept of a linguistic variable and its application to approximate reasoning, *Inform. Sci.*, 8: 199-249.
- Zimmermann, H.J., 1996. *Fuzzy set theory and its applications*, Third edition.