International Journal of Soft Computing 8 (1): 20-28, 2013
ISSN: 1816-9503

A Learning Algorithm of Fuzzy Neural Networks for Solving A System of Fuzzy Equations

A. Jafarian and S. Measoomy Nia
Department of Mathematics, Urmia Branch, Islamic Azad University, Ourmia, Iran

Abstract: Fuzzy equations system plays a major role in various areas, therefore a new method for finding a crisp solution for this system is presented. So, researchers have applied an architecture of fuzzy neural network in which the corresponding connection weights are real numbers. The proposed neural net can get a fuzzy input vector and calculates its corresponding fuzzy output. Next a learning algorithm based on the gradient descent method has been defined for adjusting the connection weights. The given approach has been illustrated by several examples with computer simulations.

Key words: System of fuzzy equations, Fuzzy Feed forward Neural Networks (FFNNs), cost function, simulation, output

INTRODUCTION

Since, many mathematical formulations of physical phenomena contain fuzzy equations and these equations are very useful for solving many problems in several applied fields like mathematical economics and optimal control theory, therefor various approaches for solving these problems have been proposed. For example, Friedman *et al.* (1998) used embedding method and suggested a general model for solving a fuzzy linear system. Theoretical aspects of a fuzzy linear system were discussed by Dubois and Prade (1980). Dehgan *et al.* (2005) employed iterative techniques for solving fully fuzzy linear system. Linear and nonlinear fuzzy systems were solved by Abbasbandy and Ezzati (2006), Abbasbandy and Alavi (2005) and Asady *et al.* (2005). In addition, one approach for indirect solution of fuzzy equations system is using Fuzzy Neural Networks (FNNs). Ishibuchi *et al.* (1995) defined a cost function for each pair of fuzzy output vector and its corresponding fuzzy target vector and proposed a learning algorithm of fuzzy neural networks with triangular and trapezoidal fuzzy weights. Hayashi *et al.* (1993) fuzzified the delta learning rule that the input-output relation of each unit was defined by the extension principle of Zadeh (1975). Abbasbandy *et al.* (2008) presented a structure of feed-forward fuzzy neural networks to find a real root of a fuzzy polynomials system by introducing a new learning algorithm.

In this study, researchers want to construct a new algorithm by using fuzzy neural networks to obtain a real solution of fuzzy equations system (if exist). For this aim, researchers apply a three-layer feed-forward neural network with fuzzy inputs, fuzzy target and crisp connection weights. Researchers consider the coefficients of fuzzy equations and the right hand fuzzy numbers of the system as input signals and target output, respectively. Also, in this neural net, the input-output relation of each unit is defined by the extension principle of Zadeh (1975). The output from this neural network which is also a fuzzy number is numerically compared with target output. Next a cost function is defined that measures the difference between the fuzzy target output and corresponding actual fuzzy output. Then, the suggested neural net using a learning algorithm that based on the gradient descent method adjusts the crisp connection weights to any desired degree of accuracy.

PRELIMINARIES

In this study, the most basic used notations in fuzzy calculus are briefly introduced. Researchers started by defining the fuzzy number.

Definition 1: A fuzzy number is a fuzzy set $u: \mathbb{R}^1 \to I = [0, 1]$ which satisfies:

- u is upper semi-continuous
- $u(x) = 0$ outside some interval $[a, d]$
- There is a real numbers b, c: $a \le b \le c \le d$ for which:
  - $u(x)$ is monotonically increasing on $[a, b]$
  - $u(x)$ is monotonically decreasing on $[c, d]$
  - $u(x) = 1, b \le x \le c$

Corresponding Author: A. Jafarian, Department of Mathematics, Urmia Branch, Islamic Azad University, Ourmia, Iran

The set of all fuzzy numbers (as given by definition 1) is denoted by E^1 (Goetschel and Voxman, 1986; Nguyen, 1978).

Definition 2: A fuzzy number ν is a pair $(\underline{\nu}, \overline{\nu})$ functions $\underline{\nu}(r), \overline{\nu}(r) : 0 \le r \le 1$ which satisfy the following requirements:

- $\underline{\nu}(r)$ is a bounded monotonically increasing left continuous function
- $\overline{\nu}(r)$ is a bounded monotonically decreasing left continuous function
- $\underline{\nu}(r) \le \overline{\nu}(r) : 0 \le r \le 1$

A popular fuzzy number is the triangular fuzzy number $\nu = (a, b, c)$ with membership function:

$$\mu_\nu(x) = \begin{cases} \dfrac{x-a}{b-a} & a \le x \le b \\ \dfrac{c-x}{c-b} & b \le x \le c \\ 0 & \text{otherwise} \end{cases}$$

where, $a \le b \le c$. It's parametric form is: $[\nu]_l^\alpha = a + (b-a)\alpha$ and $[\nu]_u^\alpha = c-(c-b)\alpha$, $0 \le \alpha \le 1$

Operation on fuzzy numbers: Researchers briefly mention fuzzy number operations defined by the extension prin-ciple (Zadeh, 1975, 2005):

$$\mu_{A+B}(z) = \max\{\mu_A(x) \wedge \mu_B(y) \mid z = x+y\}$$
$$\mu_{f(Net)}(z) = \max\{\mu_A(x) \wedge \mu_B(y) \mid z = xy\}$$

Where:

A and B = Fuzzy numbers $\mu_*(.)$ denotes the membership function of each fuzzy number
$\wedge$ = The minimum operator
$f(x) = x$ = The activation function of output unit of the fuzzy neural network

The above operations on fuzzy numbers are numerically performed on level sets (i.e., α-cuts). For $0 < \alpha \le 1$, a α-level set of a fuzzy number A is defined as:

$$[A]^\alpha = \{x \mid \mu_A(x) \ge \alpha, x \in \mathbb{R}\}$$
$$[A]^\alpha = [[A]_l^\alpha, [A]_u^\alpha]$$

where, $[A]_l^\alpha$ and $[A]_u^\alpha$ are the lower and the upper limits of the α-level set $[A]^\alpha$, respectively.

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

$$\begin{aligned} [A]^\alpha + [B]^\alpha &= [[A]_l^\alpha, [A]_u^\alpha] + [[B]_l^\alpha, [B]_u^\alpha] \\ &= [[A]_l^\alpha + [B]_l^\alpha, [A]_u^\alpha + [B]_u^\alpha] \\ f([Net]^\alpha) &= f([Net]_l^\alpha, [Net]_u^\alpha]) = [f([Net]_l^\alpha, f([Net]_u^\alpha)] \end{aligned} \quad (1)$$

$$\begin{aligned} k[A]^\alpha &= k[[A]_l^\alpha, [A]_u^\alpha] = [k[A]_l^\alpha, k[A]_u^\alpha], \text{ if } k \ge 0 \\ k[A]^\alpha &= k[[A]_l^\alpha, [A]_u^\alpha] = [k[A]_u^\alpha, k[A]_l^\alpha], \text{ if } k < 0 \end{aligned} \quad (2)$$

For arbitrary $u = (\underline{u}, \overline{u})$ and $\nu = (\underline{\nu}, \overline{\nu})$ researchers define addition $(u+\nu)$ and multiplication by k as (Goetschel and Voxman, 1986; Nguyen, 1978):

$$\overline{(u+\nu)}(r) = \overline{u}(r) + \overline{\nu}(r)$$
$$\underline{(u+\nu)}(r) = \underline{u}(r) + \underline{\nu}(r)$$
$$\overline{(ku)}(r) = k.\overline{u}(r), \underline{(k\nu)}(r) = k.\underline{u}(r), \text{ if } k \ge 0$$
$$\overline{(ku)}(r) = k.\underline{u}(r), \underline{(k\nu)}(r) = k.\overline{u}(r), \text{ if } k < 0$$

Input-output relation of each unit: Researchers have given a short review on learning of fuzzified feed-forward neural networks with fuzzy set input signals and real number weights (FFNN2) (Ishibuchi *et al*., 1995). Consider a three-layer FFNN2 with n input units, n hidden units and one output unit. In this research, researchers assume that input vector and target output are triangular fuzzy numbers and connection weights are crisp numbers. When a fuzzy input vector $A = (A_{p1}, A_{p2}, ..., A_{pn})$ is presented to the FFNN2 then the input-output relation of each unit can be written as follows:

Input units: The input neurons make no change in their inputs. So:

$$o_{pi} = A_{pi}, \quad i = 1, 2, ..., na \quad (3)$$

Hidden units:

$$O_{pj} = f(net_{pj}),$$
$$net_{pj} = w_j.o_{pj}, \quad j = 1, 2, ..., n$$

Output unit:

$$\begin{aligned} Y_p &= f(Net_P) \\ Net_P &= \sum_{j=1}^{n} W_j.O_{pj} \end{aligned} \quad (4)$$

where, $A_{p1}, ..., A_{pn}$ are triangular fuzzy numbers and w_j and W_j are crisp weights (Fig. 1). The relations between

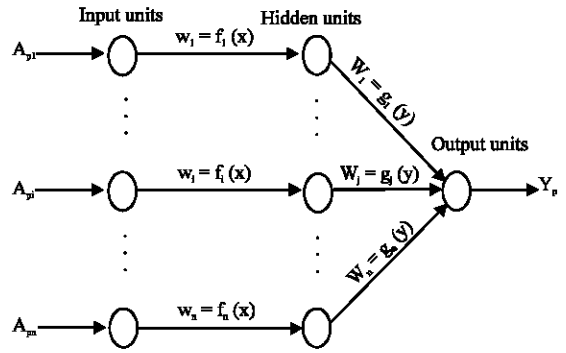

Fig. 1: The proposed FFNN2

input neurons and output neuron in Eq. 3 and 4 are defined by the extension principle (Zadeh, 1975) as in Hayashi *et al*. (1993) and Ishibuchi *et al*. (1995).

Calculation of fuzzy output: In output layer the fuzzy output neuron is numerically calculated for crisp weights and level sets. The input-output relations of the fuzzy neural network that as shown in Fig. 1 can be written for the α-level sets as follows:

Input units:

$$[o_{pi}]^{\alpha} = [A_{pi}]^{\alpha},\ i = 1, 2, ..., n \tag{5}$$

Hidden units:

$$[O_{pj}]^{\alpha} = f([net_{pj}]^{\alpha}),$$
$$[net_{pj}]^{\alpha} = w_j.[o_{pj}]^{\alpha},\ j = 1, 2, ..., n$$

Output unit:

$$[Y_p]^{\alpha} = f([Net_p]^{\alpha}),$$
$$[Net_p]^{\alpha} = \sum_{j=1}^{n} W_j.[O_{pj}]^{\alpha} \tag{6}$$

From Eq. 5 and 6, researchers can see that the α-level sets of the fuzzy output Y_p are calculated from those of the fuzzy inputs and crisp weights. From Eq. 1 and 2, the above relations are written as follows:

Input units:

$$[o_{pi}]^{\alpha} = [[o_{pi}]_l^{\alpha}, [o_{pi}]_u^{\alpha}] = [[A_{pi}]_l^{\alpha}, [A_{pi}]_u^{\alpha}]$$

Hidden units:

$$[O_{pj}]^{\alpha} = [[O_{pj}]_l^{\alpha}, [O_{pj}]_u^{\alpha}] = [f([net_{pj}]_l^{\alpha}), f([net_{pj}]_u^{\alpha})]$$

$$[net_{pj}]_l^{\alpha} = \begin{cases} w_j.[o_{pj}]_l^{\alpha} & w_j \geq 0 \\ w_j.[o_{pj}]_u^{\alpha} & w_j < 0 \end{cases}$$

$$[net_{pj}]_u^{\alpha} = \begin{cases} w_j.[o_{pj}]_u^{\alpha} & w_j \geq 0 \\ w_j.[o_{pj}]_l^{\alpha} & w_j < 0 \end{cases}$$

Output unit:

$$[Y_p]^{\alpha} = [[Y_p]_l^{\alpha}, [Y_p]_u^{\alpha}] = [f([Net_p]_l^{\alpha}), f([Net_p]_u^{\alpha})] \tag{7}$$

Where:

$$[Net_p]^{\alpha} = [[Net_p]_l^{\alpha}, [Net_p]_u^{\alpha}]$$
$$= [\sum_{j \in M} W_j.[O_{pj}]_l^{\alpha} + \sum_{j \in C} W_j.[O_{pj}]_u^{\alpha}, \sum_{j \in M} W_j.[O_{pj}]_u^{\alpha} + \sum_{j \in C} W_j.[O_{pj}]_l^{\alpha}]$$

Where:

$M = \{j \mid W_j \geq 0\}$
$C = \{j \mid W_j < 0\}$
$M \cup C = \{1, ..., n\}$

SYSTEM OF FUZZY EQUATIONS

In this study, a system of fuzzy equations is defined. Then, we will concentrate on solving this system. A system of fuzzy equations is in the following form:

$$\begin{cases} A_{11} f_{11}(x) g_{11}(y) + ... + A_{1n} f_{1n}(x) g_{1n}(y) = A_{10} \\ A_{21} f_{21}(x) g_{21}(y) + ... + A_{2n} f_{2n}(x) g_{2n}(y) = A_{20} \end{cases} \tag{8}$$

where, A_{10}, ..., A_{1n}, A_{20}, ..., A_{2n} are fuzzy numbers. In addition $f_{ij}(x)$ and $g_{ij}(y)$ (for i = 1, 2; j = 1,..., n) are real functions of the crisp variables x and y (if exist), respectively. The fuzzy numbers A_{ij} will always be real triangular fuzzy numbers.

There are two cases that must be considered. At first, researchers suppose that the functions $f_{ij}(x)$ and $g_{ij}(y)$ in both equations of the above system are equal. In other words, $f_{1j}(x) = f_{2j}(x)$ and $g_{1j}(y) = g_{2j}(y)$ (for j = 1,..., n). In next case, researchers assume that $f_{ij}(x)$ and $g_{ij}(y)$ are arbitrary real functions.

First case: Let the real functions $f_{ij}(x)$ and $g_{ij}(y)$ in both equations of the fuzzy system (8) are equal. Now, researchers want to update the crisp connection weights w_j and W_j using learning algorithm that is expressed in the following.

Cost function: Let A_{p0} be fuzzy target output corresponding to the fuzzy input vector $A_p = (A_{p1}, ..., A_{pn})$. Now, researchers intend to introduce how to deduce a learning algorithm to update the crisp weights w_j and W_j such as $w_j = f_{1j}(x)$ and $W_j = g_{1j}(y)$. At first, we define a cost function for α-level sets of fuzzy output Y_p (p = 1, 2) and it's corresponding target output A_{p0} as follows:

$$e_p^{\alpha} = e_{pl}^{\alpha} + e_{pu}^{\alpha} \tag{9}$$

Where:

$$e_{pl}^{\alpha} = \alpha.\frac{([A_{p0}]_l^{\alpha} - [Y_p]_l^{\alpha})^2}{2} \quad (10)$$

$$e_{pu}^{\alpha} = \alpha.\frac{([A_{p0}]_u^{\alpha} - [Y_p]_u^{\alpha})^2}{2} \quad (11)$$

In the cost function (9) e^{α}_{pl} and e^{α}_{pu} can be viewed as the squared errors for the lower limits and the upper limits of the α-level sets of the fuzzy output Y_p and target output A_{p0}, respectively. Then, the cost function for the input-output pair $\{A_P; A_{p0}\}$ is obtained as (Ishibuchi *et al*., 1995):

$$e_p = \sum_{\alpha} e_p^{\alpha} \quad (12)$$

Learning algorithm of the FFNN2: Let real quantities x_0 and y_0 are initialized at random values for variables x and y, respectively. Now, first researchers adjust the parameters x_0 and y_0, then will be update connection weights w_j and W_j using of x_0 and y_0, respectively. For crisp parameter y_0 adjust rule can be written as follows:

$$y_0(t+1) = y_0(t) + \Delta y_0(t) \quad (13)$$

$$\Delta y_0(t) = -\eta\frac{\partial e_p^{\alpha}}{\partial y_0} + \gamma\Delta y_0(t-1) \quad (14)$$

Where:

t = The number of adjustments
η = The learning rate
γ = The momentum term constant

Thus, the problem is calculation of derivative $\partial e_p^{\alpha}/\partial y_0$ in Eq. 14. The derivative $\partial e_p^{\alpha}/\partial y_0$ can be calculated from the cost function e_p^{α} using the input-output relation of the fuzzy neural network for the α-level sets in Eq. 6 and 7. Researchers calculate $\partial e_p^{\alpha}/\partial y_0$ as follows:

$$\frac{\partial e_p^{\alpha}}{\partial y_0} = \frac{\partial e_{pl}^{\alpha}}{\partial y_0} + \frac{\partial e_{pu}^{\alpha}}{\partial y_0} \quad (15)$$

Thus, the problem is to calculate the derivatives $\partial e^{\alpha}_{pl}/\partial y_0$ and $\partial e^{\alpha}_{pu}/\partial y_0$. Since, these two derivatives can be calculated in the same manner, only researchers show the calculation of $\partial e^{\alpha}_{pl}/\partial y_0$. Therefore, researchers have:

$$\frac{\partial e_{pl}^{\alpha}}{\partial y_0} = \frac{\partial e_{pl}^{\alpha}}{\partial [Y_p]_l^{\alpha}}.\frac{\partial [Y_p]_l^{\alpha}}{\partial [Net_p]_l^{\alpha}}.\frac{\partial [Net_p]_l^{\alpha}}{\partial y_0}$$

$$= -\alpha.([A_{p0}]_l^{\alpha} - [Y_p]_l^{\alpha}).\frac{\partial [Net_p]_l^{\alpha}}{\partial y_0}$$

Where:

$$\frac{\partial [Net_p]_l^{\alpha}}{\partial y_0} = \sum_{j=1}^{n}\left(\frac{\partial [Net_p]_l^{\alpha}}{\partial W_j}.\frac{\partial W_j}{\partial_{y0}}\right)$$

If $W_j \geq 0$:

$$\frac{\partial [Net_p]_l^{\alpha}}{\partial W_j} = [O_{pj}]_l^{\alpha}$$

Otherwise:

$$\frac{\partial [Net_p]_l^{\alpha}}{\partial W_j} = [O_{pj}]_u^{\alpha}$$

Consequently:

$$\frac{\partial e_p^{\alpha}}{\partial y_0} = -\alpha.\sum_{j\in M}\{[([A_{p0}]_l^{\alpha} - [Y_p]_l^{\alpha}).[O_{pj}]_l^{\alpha} + ([A_{p0}]_u^{\alpha} - [Y_p]_u^{\alpha}).[O_{pj}]_u^{\alpha}].(g'_{1j}(y)\big|_{y=y_0(t)}\} - \alpha.\sum_{j\in C}\{[([A_{p0}]_l^{\alpha} - [Y_p]_l^{\alpha}).[O_{pj}]_l^{\alpha} + ([A_{p0}]_u^{\alpha} - [Y_p]_u^{\alpha}).[O_{pj}]_l^{\alpha}].(g'_{1j}(y)\big|_{y=y_0(t)}\}$$

Where:

M = $\{j|W_j \geq 0\}$
C = $\{j|W_j < 0\}$

After adjusting y_0 by Eq. 13 and 14, the connection weights W_j (for j = 1, ..., n) are updated with the FFNN2 model as follows:

$$W_j(t+1) = g_{1j}(y_0(t+1)), j = 1,..., n \quad (16)$$

The crisp connection weight w_j to the jth hidden unit is updated in the same manner as the parameter weight W_j. Researchers first adjust the parameter x_0 and then will update the connection weight w_i as following:

$$x_0(t+1) = x_0(t) + \Delta x_0(t),$$

$$\Delta x_0(t) = -\eta\frac{\partial e_p^{\alpha}}{\partial x_0} + \gamma\Delta x_0(t-1)$$

Similarly, the derivative $\partial e_p^{\alpha}/\partial x_0$ can be calculated from the cost function e_p^{α} for the α-level sets in Eq. 6 and 7:

$$\frac{\partial e_p^{\alpha}}{\partial x_0} = \frac{\partial e_{pl}^{\alpha}}{\partial x_0} + \frac{\partial e_{pu}^{\alpha}}{\partial x_0}$$

Where:

$$\frac{\partial e_{pl}^{\alpha}}{\partial x_0} = \frac{\partial e_{pl}^{\alpha}}{\partial [Y_p]_l^{\alpha}}.\sum_{j=1}^{n}\left(\frac{\partial [Net_p]_l^{\alpha}}{\partial w_j}.\frac{\partial w_j}{\partial x_0}\right)$$

$$\frac{\partial e_{pu}^{\alpha}}{\partial x_0} = \frac{\partial e_{pu}^{\alpha}}{\partial [Y_p]_u^{\alpha}}.\sum_{j=1}^{n}\left(\frac{\partial [Net_p]_u^{\alpha}}{\partial w_j}.\frac{\partial w_j}{\partial x_0}\right)$$

Therefore:

$$\frac{\partial e_{pl}^{\alpha}}{\partial x_0} = -\alpha.([A_{p0}]_l^{\alpha}-[Y_p]_l^{\alpha}).\sum_{j=1}^{n}(\delta_{plj}^{\alpha}.f_j'(x)\big|x = x_0(t))$$

$$\frac{\partial e_{pu}^{\alpha}}{\partial x_0} = -\alpha.([A_{p0}]_u^{\alpha}-[Y_p]_u^{\alpha}).\sum_{j=1}^{n}(\delta_{puj}^{\alpha}.f_j'(x)\big|x = x_0(t))$$

If $W_j \geq 0$:

$$\delta_{plj}^{\alpha} = W_j.\frac{\partial[O_{pj}]_l^{\alpha}}{\partial w_j} \text{ and } \delta_{puj}^{\alpha} = W_j.\frac{\partial[O_{pj}]_u^{\alpha}}{\partial w_j}$$

Otherwise:

$$\delta_{plj}^{\alpha} = W_j.\frac{\partial[O_{pj}]_u^{\alpha}}{\partial w_j} \text{ and } \delta_{puj}^{\alpha} = W_j.\frac{\partial[O_{pj}]_l^{\alpha}}{\partial w_j}$$

If $w_j \geq 0$:

$$\frac{\partial[O_{pj}]_l^{\alpha}}{\partial w_j} = [A_{pj}]_l^{\alpha} \text{ and } \frac{\partial[O_{pj}]_u^{\alpha}}{\partial w_j} = [A_{pj}]_u^{\alpha}$$

Otherwise:

$$\frac{\partial[O_{pj}]_l^{\alpha}}{\partial w_j} = [A_{pj}]_u^{\alpha} \text{ and } \frac{\partial[O_{pj}]_u^{\alpha}}{\partial w_j} = [A_{pj}]_l^{\alpha}$$

Finally, the crisp weight w_j to the ith hidden unit can be updated as following:

$$w_j(t+1) = f_{1j}(x_0(t+1)),\ j = 1,...,n \qquad (17)$$

Now let us assume that input-output pair $\{A_p; A_{p0}\}$ where $A_p = (A_{p1},..., A_{pn})$ are given as training data and also m values of α-level sets (i.e., $\alpha_1, \alpha_2, ..., \alpha_m$) are used for the learning of fuzzy neural network. Then, the learning algo-rithm can be summarized as follows.

Learning algorithm

Step 1: $\eta>0$, $\gamma>0$, $E_{max}>0$ are chosen. Then crisp quantities x_0 and y_0 are initialized at random values.

Step 2: Let t = 0 where t is the number of iterations of the learning algorithm. Then, the running error E is set to 0.

Step 3: Calculate the crisp connection weights as follows: $w_i(t) = f_i(x_0)$ and $W_i(t) = g_i(y_0)$, i = 1,..., n.

Step 4: Let t = t + 1. Repeat step 5 for $\alpha = \alpha_1, ..., \alpha_m$.

Step 5: Repeat the following procedures for p = 1, 2.

Forward calculation: Calculate the α-level set of the fuzzy output Y_p by presenting the α-level set of the fuzzy input vector A_p.

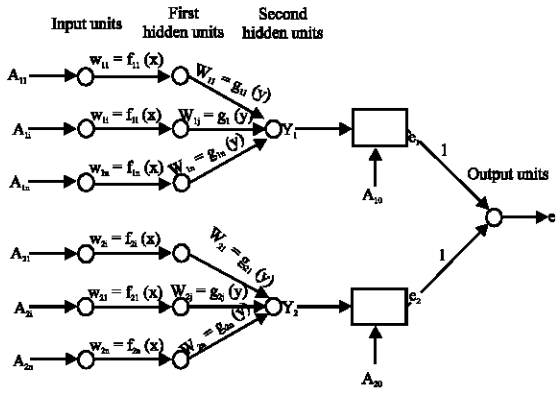

Fig. 2: The proposed FFNN2 for case 2

Back-propagation: Adjust crisp parameters x_0 and y_0 using the cost function Eq. 9 for the α-level sets of the fuzzy output Y_p and the target output A_{p0}. Then, update crisp connection weights as has been described.

Step 6: Cumulative cycle error is computed by adding the present error to E.

Step 7: The training cycle is completed. For $E<E_{max}$ terminate the training session. If $E > E_{max}$ then E is set to 0 and researchers initiate a new training cycle by going back to step 4.

Second case: In this study researchers assume that the functions $f_{ij}(x)$ and $g_{ij}(y)$ (for i = 1, 2; j = 1,.., n) in fuzzy system (8) are arbitrary real functions. In order to find an approximate solution of this kind of fuzzy systems that supposedly has a real solution, researchers have to proposed a new architecture of FFNN2. In this method all training patterns $\{A_p; A_{p0}\}$ (for p = 1, 2) where $A_p = (A_{p1}, ..., A_{pn})$ are imported to FFNN2, together. Consequently, the suggested FFNN2 is transformed to four layer neural network with 2n input units, 2n first hidden units, two second hidden units and one output unit. A FFNN2 solution to this kind of fuzzy system is given in Fig. 2. In the suggested structure of FFNN2, input vector and output vector are fuzzy numbers and connection weighs w_i and W_j are real crisp numbers. For this case the training data can be derived from the system (8) as (A; A_0) where $A = (A_{11},..., A_{1n}, A_{21},.., A_{2n})$ and $A_0 = (A_{10}, A_{20})$.

After choosing x_0 and y_0 according to the method introduced in the study, the fuzzy outputs Y_1 and Y_2 are calculated from Eq. 5 and 7. Then, two primary vectors $W = (W_{11},..., W_{1n}, W_{21},..., W_{2n})$ and $w = (w_{11},..., w_{1n}, w_{21},..., w_{2n})$ are acquired using y_0 and x_0, respectively. Then, researchers can update the crisp weights w_{ij} and W_{ij}

(for i = 1, 2; j = 1,..., n) using learning algorithm that have been expressed in last case. Therefore, the earlier cost function to be minimized is transformed for each α-level sets as follows:

$$e^{\alpha} = e_1^{\alpha} + e_2^{\alpha} \tag{18}$$

Where:

$$e_1^{\alpha} = e_{1l}^{\alpha} + e_{1u}^{\alpha},\ e_2^{\alpha} = e_{2l}^{\alpha} + e_{2u}^{\alpha}$$

And:

$$e_{1l}^{\alpha} = \alpha.\frac{([A_{10}]_l^{\alpha} - [Y_1]_l^{\alpha})^2}{2},\ e_{1u}^{\alpha} = \alpha.\frac{([A_{10}]_u^{\alpha} - [Y_1]_u^{\alpha})^2}{2}$$
$$e_{2l}^{\alpha} = \alpha.\frac{([A_{20}]_l^{\alpha} - [Y_2]_l^{\alpha})^2}{2},\ e_{2u}^{\alpha} = \alpha.\frac{([A_{20}]_u^{\alpha} - [Y_2]_u^{\alpha})^2}{2} \tag{19}$$

Correspondingly, in this case the cost function for the input-output pair {A; A_0} is obtained as (Ishibuchi *et al*., 1995):

$$e = \sum_{\alpha} e^{\alpha} \tag{20}$$

In following, the real parameters x_0 and y_0 will be adjusted according to the algorithm that has been showed in last case. Finally, researchers will update the crisp weights w_{ij} and W_{ij} with a similar manner that was represented in Eq. 16 and 17. Researchers can write the amount of adjustment for each parameter using the cost function e^{α} as follows:

$$\Delta y_0(t) = -\eta \frac{\partial e^{\alpha}}{\partial y_0} + \gamma \Delta y_0(t-1) \tag{21}$$

$$\Delta x_0(t) = -\eta \frac{\partial e^{\alpha}}{\partial x_0} + \gamma \Delta x_0(t-1) \tag{22}$$

The derivatives in Eq. 21 and 22 can be written as follows:

$$\frac{\partial e^{\alpha}}{\partial y_0} = \frac{\partial e_1^{\alpha}}{\partial y_0} + \frac{\partial e_2^{\alpha}}{\partial y_0} \text{ and } \frac{\partial e^{\alpha}}{\partial x_0} = \frac{\partial e_1^{\alpha}}{\partial x_0} + \frac{\partial e_2^{\alpha}}{\partial x_0} \tag{23}$$

Where:

$$\frac{\partial e_1^{\alpha}}{\partial y_0} = \frac{\partial e_{1l}^{\alpha}}{\partial y_0} + \frac{\partial e_{1u}^{\alpha}}{\partial y_0},\ \frac{\partial e_2^{\alpha}}{\partial y_0} = \frac{\partial e_{2l}^{\alpha}}{\partial y_0} + \frac{\partial e_{2u}^{\alpha}}{\partial y_0}$$
$$\frac{\partial e_1^{\alpha}}{\partial x_0} = \frac{\partial e_{1l}^{\alpha}}{\partial x_0} + \frac{\partial e_{1u}^{\alpha}}{\partial x_0},\ \frac{\partial e_2^{\alpha}}{\partial x_0} = \frac{\partial e_{2l}^{\alpha}}{\partial x_0} + \frac{\partial e_{2u}^{\alpha}}{\partial x_0}$$

Since, above derivatives can be calculated in the same manner, only researchers show the calculation of $\partial e^{\alpha}_{1l}/\partial y_0$ and $\partial e^{\alpha}_{1l}/\partial x_0$. Therefore, researchers have:

$$\frac{\partial e_{1l}^{\alpha}}{\partial y_0} = -\alpha.([A_{10}]_l^{\alpha} - [Y_1]_l^{\alpha}).\frac{\partial [Net_1]_l^{\alpha}}{\partial y_0}$$

Where:

$$\frac{\partial [Net_1]_l^{\alpha}}{\partial y_0} = \sum_{j=1}^{n}\left(\frac{\partial [Net_1]_l^{\alpha}}{\partial W_{1j}}.\frac{\partial W_{1j}}{\partial y_0}\right)$$

In a similar way, researchers can write above relations for $\partial^{\alpha}_{1l}/\partial x_0$ as follows:

$$\frac{\partial e_{1l}^{\alpha}}{\partial x_0} = \frac{\partial e_{1l}^{\alpha}}{\partial [Y_1]_l^{\alpha}}.\left(\begin{array}{l}\sum_{j\in M}\left(\frac{\partial [Net_1]_l^{\alpha}}{\partial [O_{1j}]_l^{\alpha}}.\frac{\partial [O_{1j}]_l^{\alpha}}{\partial x_0}\right) + \\ \sum_{j\in C}\left(\frac{\partial [Net_1]_l^{\alpha}}{\partial [O_{1j}]_u^{\alpha}}.\frac{\partial [O_{1j}]_u^{\alpha}}{\partial x_0}\right)\end{array}\right)$$

Therefore:

$$\frac{\partial e_{1l}^{\alpha}}{\partial x_0} = -\alpha.([A_{10}]_l^{\alpha} - [Y_1]_l^{\alpha}).\sum_{j=1}^{n}(\delta_{1lj}^{\alpha}.f'_{1j}(x))\Big|x = x_{0(t)})$$

If $W_{1j} \geq 0$:

$$\delta_{1lj}^{\alpha} = W_{1j}.\frac{\partial [O_{1j}]_l^{\alpha}}{\partial w_{1j}}$$

Otherwise:

$$\delta_{1lj}^{\alpha} = W_{1j}.\frac{\partial [O_{1j}]_u^{\alpha}}{\partial w_{1j}}$$

If $w_{1j} \geq 0$:

$$\frac{\partial [O_{1j}]_l^{\alpha}}{\partial w_{1j}} = [A_{1j}]_l^{\alpha} \text{ and } \frac{\partial [O_{1j}]_u^{\alpha}}{\partial w_{1j}} = [A_{1j}]_u^{\alpha}$$

Otherwise:

$$\frac{\partial [O_{1j}]_l^{\alpha}}{\partial w_{1j}} = [A_{1j}]_u^{\alpha} \text{ and } \frac{\partial [O_{1j}]_u^{\alpha}}{\partial w_{1j}} = [A_{1j}]_l^{\alpha}$$

The above relations illustrate how the derivatives in (23) for the α-level sets can be calculated. Then, the parameters y_0 and x_0 are adjusted using Eq. 21 and 22. Finally, the weights w_{ij} and W_{ij} are updated as following:

$$W_{ij}(t+1) = g_{ij}(y_0(t+1))$$
$$w_{ij}(t+1) = f_{ij}(x_0(t+1)),\ i = 1, 2 \text{ and } j = 1,..., n$$

Numerical examples: This study presents four examples to show the behavior and properties of this new method and also illustrates cases 1 and 2. For each example, the computed values of the approximate solution are calculated over a number of iterations in addition the cost function is plotted over a number of iterations.

Example 1: Consider the following system of fuzzy equations:

$$\begin{cases}(-1,0,1)e^{x-1}(y+1)^2+(-3,0,3)\sin(x-1)\cos(y)=(-1,0,1)\\(1,2,4)e^{x-1}(y+1)^2+(2,5,6)\sin(x-1)\cos(y)=(1,2,4)\end{cases}$$

where, $x, y\in\mathbb{R}$ and the exact solutions are $x = 1$ and $y = 0$. In this example, researchers apply the proposed method to approximate the solution of this fuzzy system. The training starts by $x_0 = 0.5$, $y_0 = 0.5$, $\eta = 0.005$ and $\gamma = 0.001$. Table 1 shows the approximated solution over a number of iterations and Fig. 3 and 4 show the accuracy of the solutions x_0 (t) and y_0 (t) where t is the number of iterations.

Table 1: The approximated solutions with error analysis for example 1

t	x_0 (t)	y_0 (t)	e	t	x_0 (t)	y_0 (t)	e
1	0.59976	0.54282	18.5871	582	0.98819	0.017880	0.0254800
2	0.63228	0.52656	15.5768	583	0.98931	0.016186	0.0208882
3	0.65463	0.50065	14.6652	284	0.99032	0.014653	0.0171223
4	0.67568	0.47312	13.5173	285	0.99124	0.013264	0.0140342
5	0.69641	0.44524	12.3184	586	0.99207	0.012007	0.0115021
6	0.71677	0.41737	11.1192	587	0.99282	0.010868	0.0094263

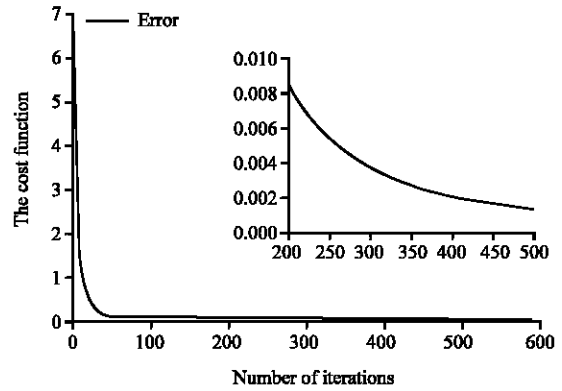

Fig. 3: The cost function for example 1 over the number of iterations

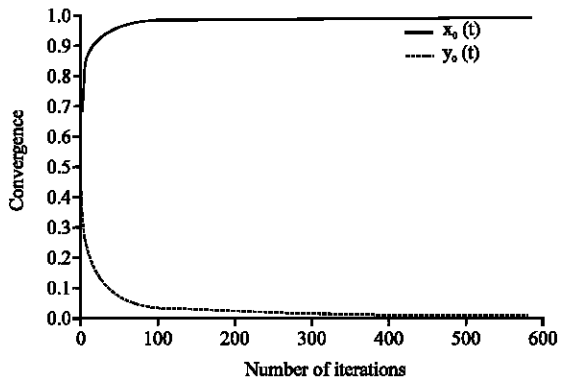

Fig. 4: Convergence of the approximated solution for example 1

Example 2: Let fuzzy system:

$$\begin{cases}(1,2,3)e^{x}y+(2,3,4)\sin(x)\cos(y)=(1,2,3)\\(2,3,4)x^2\sin(y-1)+(3,4,5)(x+1)^3y^3=(3,4,5)\end{cases}$$

with the exact solutions $x = 0$ and $y = 1$. Researchers trained the fuzzy neural network as described in last example. Before starting calculations, researchers assume that $x_0 = 0.5$, $y_0 = 0.5$, $\eta = 0.002$ and $\gamma = 0.002$. Numerical result can be found in Table 2. Fig. 5 and 6 show the accuracy of the solutions x_0 (t) and y_0 (t) where t is the number of iterations.

Table 2: The approximated solutions with error analysis for example 2

t	x_0 (t)	y_0 (t)	e	t	x_0 (t)	y_0 (t)	e
1	0.45681	0.68413	31.2999	27	0.063141	0.93859	0.1581700
2	0.38927	0.71539	14.0371	28	0.060363	0.94115	0.1429990
3	0.33769	0.74213	9.61541	29	0.057752	0.94356	0.1295570
4	0.29724	0.76503	6.86418	30	0.055294	0.94584	0.1176100
5	0.26478	0.78474	5.07066	31	0.052976	0.94800	0.1069610
6	0.23818	0.80183	3.85340	32	0.050788	0.95005	0.0974454

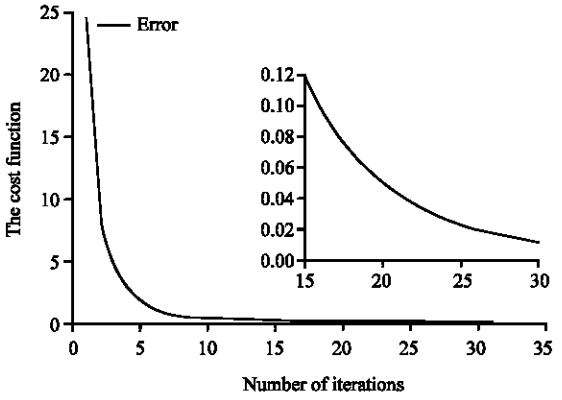

Fig. 5: The cost function for example 2 over the number of iterations

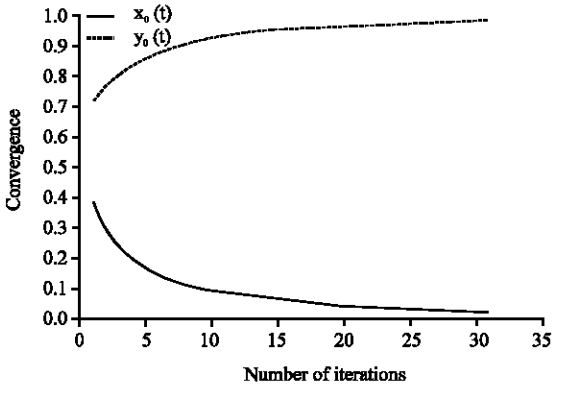

Fig. 6: Convergence of the approximated solution for example 2

Example 3: Consider the following fuzzy system:

$$\begin{cases}(2,3,5)\cos(x^2)\cos^3(y)+(1,3,7)\tan^3\left(\frac{\Pi}{4}(x+1)\right)\ln^{e(y+1)}=(3,6,12)\\(3,7,8)\cos(x^2)\cos^3(y)+(0,2,3)\tan^3\left(\frac{\Pi}{4}(x+1)\right)\ln^{e(y+1)}=(3,9,11)\end{cases}$$

with the exact solutions x = 0 and y = 0. Researchers trained the fuzzy neural network as described in last example. Before starting calculations, researchers assume that $x_0 = 0.5$, $y_0 = 1$, $\eta = 0.002$ and $\gamma = 0.002$. Numerical result can be found in Table 3. Figure 7 and 8 show the accuracy of the solutions x_0 (t) and y_0 (t) where t is the number of iterations.

Table 3: The approximated solutions with error analysis for example 3

t	x_0 (t)	y_0 (t)	e	t	x_0 (t)	y_0 (t)	e
1	0.05946	0.79148	453.8362	492	-0.015866	0.026449	0.001018941
2	0.01405	0.51549	16.81881	493	-0.015854	0.026440	0.001015053
3	-0.00687	0.43755	4.057131	494	-0.015842	0.026432	0.001011188
4	-0.01715	0.40156	1.516875	495	-0.015830	0.026424	0.001007345
5	-0.02271	0.38069	0.774076	496	-0.015818	0.026415	0.001003524
6	-0.02587	0.36686	0.488321	497	-0.015806	0.026407	0.000999724

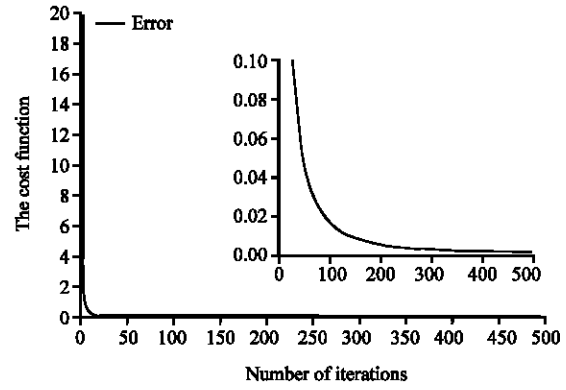

Fig. 7: The cost function for example 3 over the number of iterations

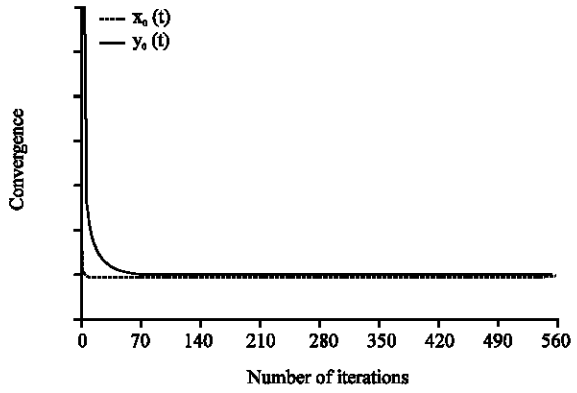

Fig. 8: Convergence of the approximated solution for example 3

Example 4: Consider the following system of fuzzy equations:

$$\begin{cases}(0,1,2)x^3\cos(y)+(-3,-2,-1)\ln^{ex}e^y=(-3,-1,1)\\(1,2,3)\cos(x-1)e^{2y}+(-2,-1,0)x^2(y+1)^2=(-1,1,3)\end{cases}$$

with the exact solution x = 1 and y = 0. Before starting calculations, researchers assume that $x_0 = 0.75, y_0 = 0.75, \eta = 0.002$ and $\gamma = 0.002$ (Table 4, Fig. 9 and 10).

Table 4: The approximated solutions with error analysis for example 4

t	x_0 (t)	y_0 (t)	e	t	x_0 (t)	y_0 (t)	e
1	0.74529	0.062583	64.2548	492	0.97748	-0.0057331	0.1139190
2	0.81180	0.016805	4.99624	493	0.98633	-0.0041192	0.0444066
3	0.86711	-0.001251	2.65779	494	0.99176	-0.0028300	0.0168285
4	0.91029	-0.007562	1.36118	495	0.99505	-0.0018854	0.0168280
5	0.94181	-0.008572	0.64229	496	0.99704	-0.0012289	0.0023177
6	0.96341	-0.007444	0.27940	497	0.99822	-0.0007881	0.0008522

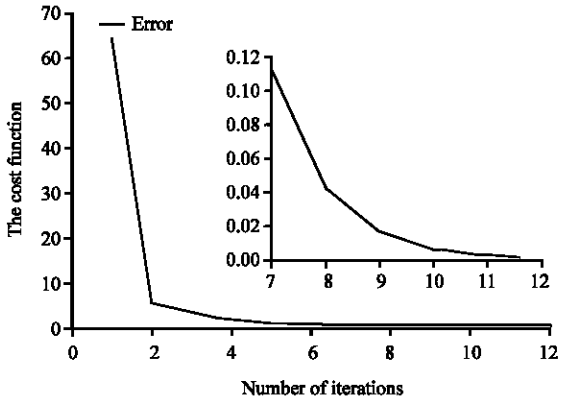

Fig. 9: The cost function for example 4 over the number of iterations

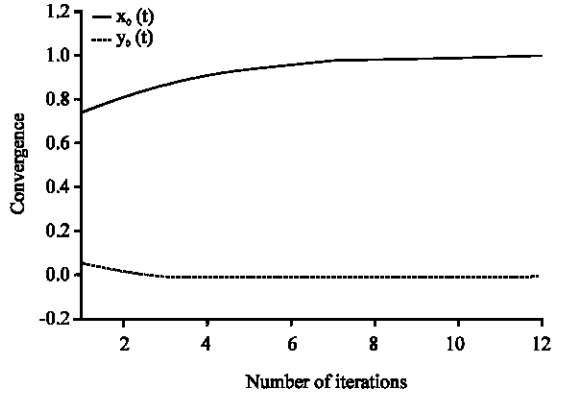

Fig. 10: Convergence of the approximated solution for example 4

CONCLUSION

In this research, a new architecture of feed-forward neural networks has been proposed to approximate solution of fuzzy equations system. Presented FFNN2 in this study is a numerical method for calculating unknown variables in the given system. To get the best approximating solution of the equation, number of iterations must be chosen large enough. With the availability of this methodology, now it will be possible to investigate the approximate solution of other kinds of fuzzy systems. The analyzed examples illustrate the ability and reliability of the present method. The obtained solutions, in comparison with exact solutions admit a remarkable accuracy.

REFERENCES

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.

Abbasbandy, S. and M. Alavi, 2005. A method for solving fuzzy linear systems. Iran. J. Fuzzy Syst., 2: 37-43.

Abbasbandy, S. and R. Ezzati, 2006. Newton's method for solving a system of fuzzy nonlinear equations. Appl. Math. Comput., 175: 1189-1199.

Alefeld, G. and J. Herzberger, 1983. Introduction to Interval Computations. Aca-demic Press, New York, USA.

Asady, B., S. Abbasbandy and M. Alavi, 2005. Fuzzy general linear systems. Applied Math. Comput., 169: 34-40.

Dehgan, M., B. Hashemi and M. Ghatee, 2005. Solution of the fully fuzzy linear systems using iterative techniques. Chaos Solitons Fract., 26: 835-843.

Dubois, D. and H. Prade, 1980. Systems of linear fuzzy constraints. Fuzzy Sets Syst., 3: 37-48.

Friedman, M., M. Ming and A. Kandel, 1998. Fuzzy linear systems. Fuzzy Sets Syst., 96: 201-209.

Goetschel, R. and W. Voxman, 1986. Elementary calculus. Fuzzy Sets Syst., 18: 31-43.

Hayashi, Y., J.J. Buckley and E. Czogala, 1993. Fuzzy neural network with fuzzy signals and weights. Int. J. Intell. Syst., 8: 527-537.

Ishibuchi, H., K. Kwon and H. Tanaka, 1995. A learning of fuzzy neural networks with triangular fuzzy weghts. Fuzzy Sets Syst., 71: 277-293.

Nguyen, H.T., 1978. A note on the extension principle for fuzzy sets. J. Math. Anal. Applied, 64: 369-380.

Zadeh, L.A., 1975. The concept of a linguistic variable and its application to approximate reasoning-III. Inform. Sci., 9: 43-80.

Zadeh, L.A., 2005. Toward a Generalized Theory of Uncertainty (GTU) an outline. Inform. Sci., 172: 1-40.