Journal of Advances in Computer Research
Quarterly ISSN: 2008-6148
Sari Branch, Islamic Azad University, Sari, I.R.Iran
(Vol. 3, No. 4, November 2012), Pages: 33-45
www.jacr.iausari.ac.ir

Solving Fuzzy Equations Using Neural Nets with a New Learning Algorithm

Ahmad Jafarian[1]✉, Safa Measoomy nia[1], Raheleh Jafari[2]

(1)Department of Mathematics, Urmia Branch, Islamic Azad University, Urmia, Iran
(2)Department of Mathematics, science and research Branch, Islamic Azad University, Arak, Iran

jafarian5594@yahoo.com; measoomy@yahoo.com; jafari3339@yahoo.com

Received: 2012/04/17; Accepted: 2012/10/06

Abstract

Artificial neural networks have the advantages such as learning, adaptation, fault-tolerance, parallelism and generalization. This paper mainly intends to offer a novel method for finding a solution of a fuzzy equation that supposedly has a real solution. For this scope, we applied an architecture of fuzzy neural networks such that the corresponding connection weights are real numbers. The suggested neural net can adjust the weights using a learning algorithm that based on the gradient descent method. The proposed method is illustrated by several examples with computer simulations.

Keywords: *Fuzzy equations, Fuzzy feed-forward neural network (FFNN), Cost function, Learning algorithm*

1. Introduction

Fuzzy equations are very useful for solving many problems in several applied fields like mathematical economics and optimal control theory, because many mathematical formulations of physical phenomena contain these kinds of equations. Therefore, various approaches for solving these problems have been proposed. One approach to indirect solution is using fuzzy neural networks (FNNs). In recent years the FNN model has been rapidly developed and put into use in a wide variety fields [9, 10, 21]. FNNs are simplified models of the biological nervous system and therefore have drawn their motivation from the kind of computing performed by a human brain. FNNs are in general capable of close approximation of the prediction model without the need of its explicit (mathematical) formulation in contrast to statistical approaches.

First time, Buckley in [8] applied a structure of FNNs for solving fuzzy equations. Ishibuchi et al. [13] defined a cost function for every pair of fuzzy output vector and its corresponding fuzzy target vector and also designed a learning algorithm of fuzzy neural networks with triangular and trapezoidal fuzzy weights. Hayashi et al. [12] fuzzified the delta rule and summarized much of the work in fuzzy neural networks. Buckley and Eslami [7] employed neural nets to solve fuzzy problems with both real and complex fuzzy numbers. Moreover, Linear and nonlinear fuzzy equations have been solved in [1, 2, 3, 6]. Jafarian et al. [16] proposed a numerical scheme to solve fuzzy linear volterra integral equations system. The topic of numerical solution of fuzzy

polynomials by FNN investigated by Abbasbandy et al. [4], consist of finding solution to polynomials like $a_1x+...+a_nx_n=a_0$ where $x\in R$ and $a_0,a_1,...,a_n$ are fuzzy numbers. Jafarian and Jafari [15] applied fuzzy feed-back neural network method for approximation of the crisp solution of dual fuzzy polynomials. We refer the reader to [20, 22] for more information on fuzzy polynomials.

The objective of this paper is primarily to design a new model based on FNNs for approximate solution of fuzzy equations. In this work, an architecture of FFNN2 (fuzzy feed-forward neural network with fuzzy set input signals, fuzzy output signal and real number weights) equivalent to fuzzy equation of the form $A_1f_1(x)+...+A_nf_n(x)=A_0$ is built, where $A_0,A_1,...,A_n$ are fuzzy numbers and $f_i(x)$ (for $i = 1, ..., n$) are real functions. The proposed neural network has two layers that the input-output relation of each unit is defined by the extension principle of Zadeh [23]. The coefficients of the fuzzy equation and the right hand fuzzy number are considered as input signals and target output, respectively. The output from the neural network which is also a fuzzy number, is numerically compared with the 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.

The remainder of the paper is organized according to the following outline: In Section 2, some basic definitions are presented. In section 3, the fuzzy equation is briefly described. In this section, we describe how to find a real solution of the fuzzy equation by using FFNs. In section 4, the applicability of the method is illustrated by several examples in which the exact solution and the computed results are compared with each other. Finally, conclusion is described in section 5.

2. Preliminaries

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

Definition 1. A fuzzy number is a fuzzy $R^1 \to I=[0,1]$ such that:

i. u is upper semi-continuous,

ii. $u(x)=0$ outside some interval $[a, d]$,

iii. There are real numbers $b,c: a\le b\le c\le d$, for which:

1. $u(x)$ is monotonically increasing on $[a, b]$,
2. $u(x)$ is monotonically decreasing on $[c, d]$,
3. $u(x)=1,\ b\le x\le c$.

The set of all fuzzy numbers (as given by definition 1) is denoted by E^1 [11, 19]. An alternative definition which yields the same E^1 is given by Kaleva [17] and Ma et al. [18].

Definition 2. A fuzzy number v is a pair $(\underline{v},\overline{v})$ of function $\underline{v}(r)$ and $\overline{v}(r): 0\le r\le 1$, which satisfy the following requirements:

i. $\underline{v}(r)$ is a bounded monotonically increasing, left continuous function on (0, 1] and right continuous at 0,

ii. $\overline{v}(r)$ is a bounded monotonically decreasing, left continuous function on (0, 1] and right continuous at 0,

iii. $\underline{v}(r) \le \overline{v}(r) : 0 \le r \le 1.$

A popular fuzzy number is the triangular fuzzy number $v = (v_m, v_l, v_u)$ where v_m denotes the modal value and the real values $v_l \ge 0$ and $v_u \ge 0$ represent the left and right fuzziness, respectively. The membership function of a triangular fuzzy number is defined as follows:

$$\mu_v(x) = \begin{cases} \dfrac{x - v_m}{v_l} + 1 & , v_m - v_l \le x \le v_m, \\ \dfrac{v_m - x}{v_u} + 1 & , v_m \le x \le v_m + v_u, \\ 0 & , otherwise. \end{cases}$$

Its parametric form is:

$$\underline{v}(r) = v_m + v_l(r-1),\ \overline{v}(r) = v_m + v_u(1-r),\ 0 \le r \le 1.$$

Triangular fuzzy numbers are fuzzy numbers in *LR* representation where the reference functions *L* and *R* are linear.

2.1 Operations on fuzzy numbers

We briefly mention fuzzy number operations defined by the extension principle [23,24].

$$\mu_{A+B}(z) = \max\left\{ \mu_A(x) \wedge \mu_B(y) \middle| z = x + y \right\},$$

$$\mu_{f(Net)}(z) = \max\left\{ \mu_A(x) \wedge \mu_B(y) \middle| z = xy \right\},$$

where *A* and *B* are fuzzy numbers, $\mu_*(.)$ denotes the membership function of each fuzzy number, $\wedge$ is the minimum operator, and *f* is a continuous activation function (such as $f(x) = x$) of output unit of our 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 $\alpha -$level set of a fuzzy number *A* is defined as:

$$[A]^\alpha = \left\{ x \middle| \ \mu_A \ge (x) \ge \alpha, x \in R \right\},$$

and $[A]^0 = \overline{\cup_{\alpha \in (0,1]} [A]^\alpha}$. Since level sets of fuzzy numbers become closed intervals, we denote $[A]^\alpha$ by

$$[A]^\alpha = \left[[A]_l^\alpha, [A]_u^\alpha \right],$$

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 [5], the above operations on fuzzy numbers are written for the α -level sets as follows:

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

$$k[A]^{\alpha} = k\left[[A]_l^{\alpha}, [A]_u^{\alpha}\right] = \left[k[A]_l^{\alpha}, k[A]_u^{\alpha}\right], \quad if \quad k \geq 0,$$
$$k[A]^{\alpha} = k\left[[A]_l^{\alpha}, [A]_u^{\alpha}\right] = \left[k[A]_l^{\alpha}, k[A]_u^{\alpha}\right], \quad if \quad k < 0. \tag{2}$$

For arbitrary $u = (\underline{u}, \overline{u})$ and $v = (\underline{v}, \overline{v})$ we define addition $(u + v)$ and multiplication by k as [11, 19]:

$$\overline{(u+v)}(r) = \overline{u}(r) + \overline{v}(r),$$
$$\underline{(u+v)}(r) = \underline{u}(r) + \underline{v}(r),$$
$$\overline{(ku)}(r) = k.\overline{u}(r), \ \underline{(kv)}(r) = k.\underline{u}(r), \quad if \quad k \geq 0,$$
$$\overline{(ku)}(r) = k.u(r), \ \underline{(kv)}(r) = k.\overline{u}(r), \quad if \quad k < 0.$$

3. Fuzzy equation

We are interested in finding a real solution of the fuzzy equation (if exist) in general form

$$A_1 f_1(x) + \ldots + A_n f_n(x) = A_0, \tag{3}$$

where $A_i \in E^1$ and $f_i(x)$ (for $i = 1, \ldots, n$) are real functions.

For getting an approximate solution, an architecture of feed-forward neural network equivalent to Eq. (3) is built. The network is shown in Figure 1.

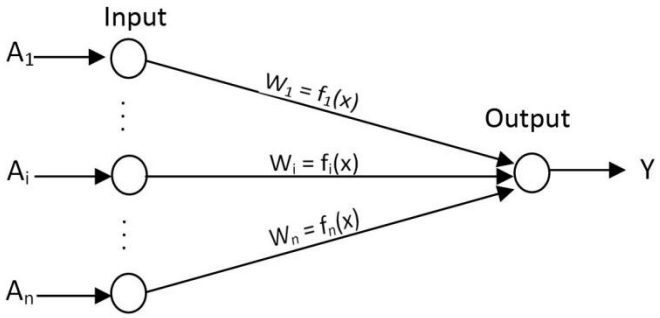

Figure 1. The proposed neural network.

3.1 Input-output relation of each unit

Consider a two-layer FFNN2 with n input neurons and one output neuron. It is clear that the input vector, the target output are triangular fuzzy numbers and connection weights are crisp numbers. When a fuzzy input vector $A = (A_1, A_2, \ldots, A_n)$ is presented to the FFNN2, then the input-output relation of each unit can be written as follows (see Figure 1):

Input units:

The input neurons make no change in their inputs, so:

$$O_i = A_i, \ i = 1, 2, \ldots, n. \tag{4}$$

Output unit:
$Y = f(Net)$,

$$Net = \sum_{i=1}^{n} (w_i . O_i), \tag{5}$$

where A_i is a triangular fuzzy number and w_i is a crisp connection weight. The relations between the input neurons and the output neuron in Eqs. (4)-(5) are defined by the extension principle [23] as in Hayashi et al. [12] and Ishibuchi et al. [14].

3.2 Calculation of fuzzy output

The fuzzy output from of neuron in the second layer is numerically calculated for crisp weights and level sets of fuzzy inputs. The input-output relations of the fuzzy neural network as shown in Figure 1 can be written for the α-level sets as follows:
Input units:

$$[O_i]^{\alpha} = [A_i]^{\alpha}, \quad i = 1, ..., n. \tag{6}$$

Output unit:
Let f be an one-to-one activation function. Now we have:

$$\begin{aligned} [Y]^{\alpha} &= f([Net]^{\alpha}), \\ [Net]^{\alpha} &= \sum_{i=1}^{n} (w_i . [O_i]^{\alpha}). \end{aligned} \tag{7}$$

From Eqs. (6)-(7), we can conclude that the α-level sets of the fuzzy output Y are calculated from those of the fuzzy inputs and crisp weights. From Eqs. (1)-(2), the above relations are transformed to following form:
Input units:

$$[O_i]^{\alpha} = \left[[O_i]_l^{\alpha}, [O_i]_u^{\alpha} \right] = \left[[A_i]_l^{\alpha}, [A_i]_u^{\alpha} \right], \quad i = 1, \ldots, n.$$

Output unit:

$$[Y]^{\alpha} = \left[[Y]_l^{\alpha}, [Y]_u^{\alpha} \right] = \left[f([Net]_l^{\alpha}), f([Net]_u^{\alpha}) \right], \tag{8}$$

where

$$[Net]^{\alpha} = \left[[Net]_l^{\alpha}, [Net]_u^{\alpha} \right] =$$

$$\left[\sum_{i \in M} (w_i . [O_i]_l^{\alpha}) + \sum_{i \in C} (w_i . [O_i]_u^{\alpha}), \sum_{i \in M} (w_i . [O_i]_u^{\alpha}) + \sum_{i \in C} (w_i . [O_i]_l^{\alpha}), \right],$$

where $M = \{ i \mid w_i \geq 0 \}, C = \{ i \mid w_i < 0 \}$ and $M \cup C = \{1, \ldots, n\}$.

Lemma 1. Let fuzzy equation $\sum_{i=1}^{n} A_i . f_i(x) = A_0$ has a solution for real crisp number x, then

$D \neq \varphi$ where $D = \cap_{i=1}^{n} \; domain \; (f_i(x))$.

Proof. Let $x_0 \in R$ be a solution of Eq. (3). Consequently, we can write:

$$A_1 f_1(x_0) + \ldots + A_n f_n(x_0) = A_0,$$

and it follows from the above relation that $f_i(x_0)$ exists. It is clear that $x_0 \in domain\ (f_i(x))$ and consequently $x_0 \in \cap_{i=1}^{n}\ domain\ (f_i(x)) \neq \phi$

Corollary 1. The necessary condition for existence solution of Eq. (3) is $D \neq \phi$.

An architecture of FFNN2 solution to Eq. (3) has been given in Figure 1. The modelling scheme is designed with the simple and versatile fuzzy neural network architecture. Now let A_0 be the target output corresponding to the fuzzy input vector $A = (A_1, ..., A_n)$. We want to define a cost function for the α-level sets of the fuzzy output Y and the corresponding target output A_0:

$$e^{\alpha} = e_l^{\alpha} + e_u^{\alpha}, \tag{9}$$

where

$$e_l^{\alpha} = \frac{([A_0]_l^{\alpha} - [Y]_l^{\alpha})^2}{2},$$

$$e_u^{\alpha} = \frac{([A_0]_u^{\alpha} - [Y]_u^{\alpha})^2}{2}.$$

In the cost function (9), e_l^{α} and e_u^{α} 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 and target output A_0, respectively. Now the cost function for the input-output pair $\{A; A_0\}$ is obtained as:

$$e = \sum_{\alpha} e^{\alpha}. \tag{10}$$

3.3 Learning algorithm of the FFNN2

Let a real quantity x_0 is initialized at random value for variable x. The main aim of this subdivision is to offer an algorithm for adjusting the parameter x_0 and connection weight w_i. For real parameter x_0 adjustment rule can be written as follows:

$$x_0(t+1) = x_0(t) + \Delta x_0(t), \tag{11}$$

$$\Delta x_0(t) = -\eta \cdot \frac{\partial e^{\alpha}}{\partial x_0} + \gamma \cdot \Delta x_0(t-1), \tag{12}$$

where t is the number of adjustments, η is the learning rate and γ is the momentum term constant. Thus our problem is to calculate the derivative $\frac{\partial e^{\alpha}}{\partial x_0}$ in (12). The given derivative can be calculated from the cost function e^{α} and by using the input-output relations (6)-(7). The derivative $\frac{\partial e^{\alpha}}{\partial x_0}$ can be calculated as follows:

$$\frac{\partial e^{\alpha}}{\partial x_0} = \frac{\partial e_l^{\alpha}}{\partial x_0} + \frac{\partial e_u^{\alpha}}{\partial x_0}, \tag{13}$$

where

$$\frac{\partial e_l^\alpha}{\partial x_0} = \frac{\partial e_l^\alpha}{\partial [Y]_l^\alpha} \cdot \frac{\partial [Y]_l^\alpha}{\partial [Net]_l^\alpha} \cdot \frac{\partial [Net]_l^\alpha}{\partial x_0},$$

$$\frac{\partial e_u^\alpha}{\partial x_0} = \frac{\partial e_u^\alpha}{\partial [Y]_u^\alpha} \cdot \frac{\partial [Y]_u^\alpha}{\partial [Net]_u^\alpha} \cdot \frac{\partial [Net]_u^\alpha}{\partial x_0},$$

and

$$\frac{\partial [Net]_l^\alpha}{\partial x_0} = \sum_{i=1}^{n} \left(\frac{\partial [Net]_l^\alpha}{\partial w_i} \cdot \frac{\partial w_i}{\partial x_0}\right) = \sum_{i \in M} ([A_i]_l^\alpha \cdot f_i'(x_0)) + \sum_{i \in C} ([A_i]_u^\alpha \cdot f_i'(x_0)),$$

$$\frac{\partial [Net]_u^\alpha}{\partial x_0} = \sum_{i=1}^{n} \left(\frac{\partial [Net]_u^\alpha}{\partial w_i} \cdot \frac{\partial w_j}{\partial x_0}\right) = \sum_{i \in M} ([A_i]_u^\alpha \cdot f_i'(x_0)) + \sum_{i \in C} ([A_i]_l^\alpha \cdot f_i'(x_0)).$$

Consequently

$$\Delta x_0(t) = \eta \cdot \sum_{i \in M} \left(\left[([A_0]_l^\alpha - [Y]_l^\alpha) \cdot [A_i]_l^\alpha + ([A_0]_u^\alpha - [Y]_u^\alpha) \cdot [A_i]_u^\alpha \right] \cdot f_i'(x_0(t))) + \right.$$

$$\eta \cdot \sum_{i \in C} \left(\left[([A_0]_l^\alpha - [Y]_l^\alpha) \cdot [A_i]_u^\alpha + ([A_0]_u^\alpha - [Y]_u^\alpha) \cdot [A_i]_l^\alpha \right] \cdot f_i'(x_0(t))) + \gamma \cdot \Delta x_0(t-1), \right.$$

where $M = \{i \mid w_i \geq 0\}$, $C = \{i \mid w_i < 0\}$ and $M \cup C = \{1, \ldots, n\}$. After adjusting the parameter x_0 by using Eqs. (11)-(12), the connection weights are updated as follows:

$$w_i(t+1) = f_i(x_0(t+1)), \quad i = 1, \ldots, n.$$

Let us assume that input-output pair $\{A; A_0\}$ where $A = (A_1, \ldots, A_n)$ are given as training data and also m values of $\alpha-$level sets (i.e. $\alpha_1, \alpha_2, \ldots, \alpha_m$) are used for the learning of fuzzy neural network. Then the learning algorithm can be summarized as follows:

Learning algorithm

Step 1: $\eta > 0$, $\gamma > 0$ and $Emax > 0$ are chosen. Also crisp quantity $x_0 \in D$ is initialized at random value.

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:

$$w_i(t) = f_i(x_0(t)), \quad i = 1, \ldots, n.$$

Step 4: Let $t := t + 1$. Repeat *Step 5* for $\alpha = \alpha_1, \alpha_2, \ldots, \alpha_m$..

Step 5:

i. Forward calculation: Calculate the α-level set of the fuzzy output Y by presenting the $\alpha-$level set of the fuzzy input vector A.

ii. Back-propagation: Adjust the parameter x_0 using the cost function (9) for the α-level sets of the fuzzy output Y and the target output A_0. Then update the weights by using *Step 3*.

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

Step 7: The training cycle is completed. For $E < Emax$ terminate the training session. If $E > Emax$ then E is set to 0 and we initiate a new training cycle by going back to *Step 4*.

4. Numerical examples

To show the behavior and properties of the proposed method, four examples have been solved in this section. For each example, the computed values of the approximate solution are calculated and the cost function is plotted over a number of iterations.

Example 4.1. Consider the following fuzzy equation:

$$(1,2,3)e^{x}+(2,3,4)\sin(x)+(3,4,5)x^{3}=(1,2,3),$$

with the exact solution is $x=0$ and $x\in R$. In this example, we apply the proposed method to approximate solution of this fuzzy equation. The training pattern is as follows:

$$\{A, A_0\}=\{(1,2,3),(2,3,4),(3,4,5);(1,2,3)\}.$$

The FFNN2 is trained with three input units and single output neuron. The training starts with $x_0=1$, η = 0.001 and γ = 0.001. Table 1 shows the approximated solution over a number of iterations and Figures 2 and 3 show the accuracy of the calculated solution $x_0(t)$.

Table 1. The approximated solutions with error analysis for Example 4.1

t	$x_0(t)$	*e*	*t*	$x_0(t)$	*e*
1	0.83914	861.0975	21	0.011710	0.0522081
2	0.70215	462.5263	22	0.009246	0.0325274
3	0.58561	256.2918	23	0.0072979	0.0202522
4	0.48657	150.7682	24	0.0057581	0.0126028
5	0.40254	90.27531	25	0.004542	0.0078392
6	0.33146	55.26543	26	0.003582	0.0048745
7	0.27161	34.36732	27	0.002824	0.0030302
8	0.22148	21.59092	28	0.002226	0.0018833
9	0.17975	13.64433	29	0.001755	0.0011702
10	0.14524	8.645593	30	0.001383	0.0007271

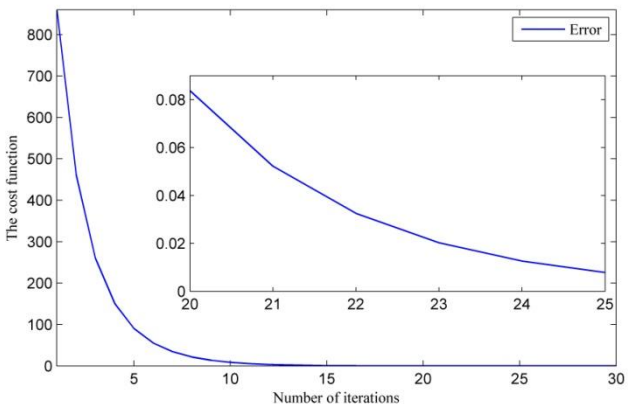

Figure 2. The cost function for Example 4.1 over the number of iterations.

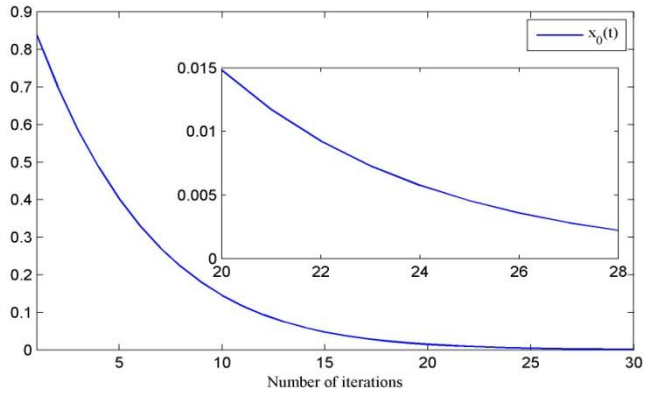

Figure 3. Convergence of the approximated solution for Example 4.1.

Example 4.2. Let fuzzy equation

$$(-1,0,1)x^2 + (-2,0,2)x^4 + (-3,0,3)\sin(x-1) = (-3,0,3),$$

with the exact solution $x=1$. Similarly, we assumed that $x_0 = 0.5$, η = 0.01 and γ = 0.01. Numerical result can be found in Table 2. Also Figures 4 and 5 show the accuracy of the solution $x_0(t)$.

Table 2. The approximated solutions with error analysis for Example 4.2

t	$x_0(t)$	e	t	$x_0(t)$	e
1	0.58610	420.5535	13	0.97840	0.0245014
2	0.66124	356.6554	14	0.98358	0.0221424
3	0.72595	303.7533	15	0.98753	0.0200111
4	0.78074	259.7104	16	0.99054	0.0180855
5	0.82634	222.8538	17	0.99283	0.0132040
6	0.86367	191.8637	18	0.99456	0.0076470
7	0.89378	165.6911	19	0.99588	0.0044163
8	0.91775	143.4962	20	0.99688	0.0025451
9	0.93663	124.6025	21	0.99764	0.0014644
10	0.95138	108.4619	22	0.99821	0.0008415

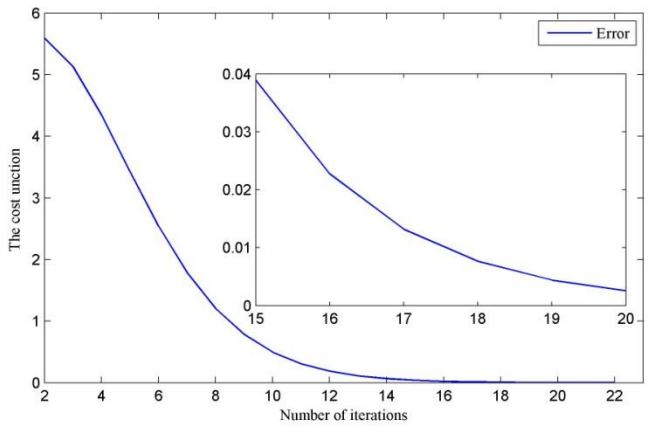

Figure 4. The cost function for Example 4.2 over the number of iterations.

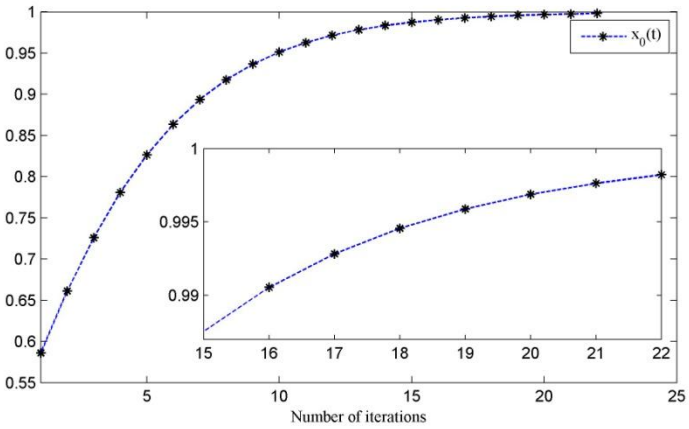

Figure 5. Convergence of the approximated solution for Example 4.2.

Example 4.3. We consider the fuzzy equation

$$(-2,-1,1)e^{x}+(1,2,3)\tan(\frac{\pi}{4}(x+1))+(3,5,6)\ln^{e(x+1)}=(2,6,10),$$

With the exact solution $x=0$. Learning is started with $x_0=-0.5$, η = 0.002 and γ = 0.002. Similarly, Numerical result can be found in Table 3. Figures 6 and 7 show the accuracy of the approximate solution $x_0(t)$.

Table 3. The approximated solutions with error analysis for Example 4.3

t	$x_0(t)$	e	***t***	$x_0(t)$	e
1	-0.38289	187.7403	33	-0.0034570	0.009737422
2	-0.30599	101.4838	34	-0.0030276	0.007466416
3	-0.25043	62.25337	35	-0.0026515	0.005725975
4	-0.20813	40.74910	36	-0.0023223	0.004391846
5	-0.17484	27.75027	37	-0.0020343	0.003368974
6	-0.14804	19.40627	38	-0.0017816	0.002584605
7	-0.12610	13.82995	39	-0.0015606	0.001983039
8	-0.10792	9.995062	40	-0.0013671	0.001521611
9	-0.09271	7.301395	41	-0.0011976	0.001167634
10	-0.07988	5.378616	42	-0.0010491	0.000896059

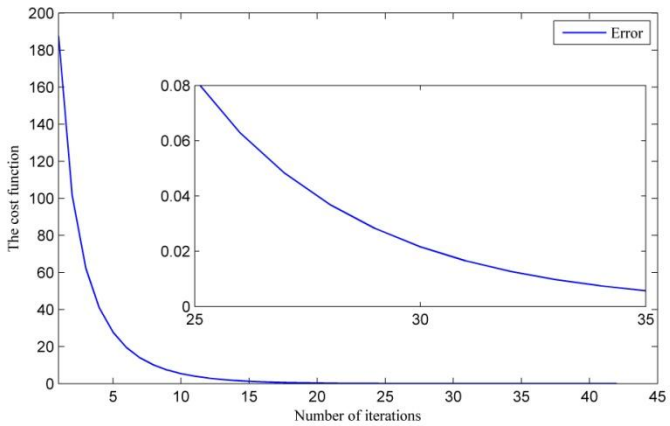

Figure 6. The cost function for Example 4.3 over the number of iterations.

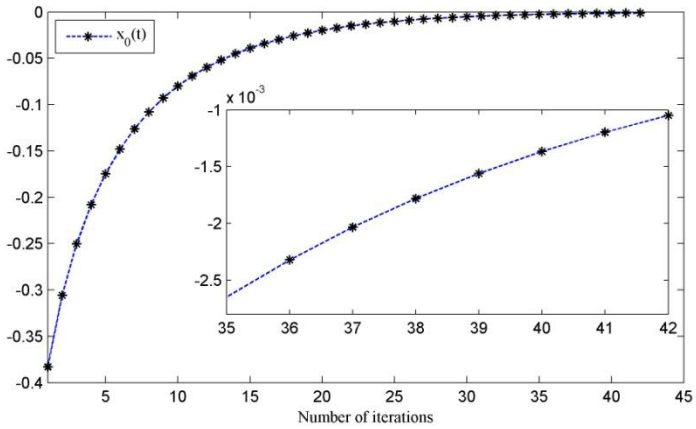

Figure 7. Convergence of the approximated solution for Example 4.3.

Example 4.4. Consider the fuzzy equation

$$(2,4,6)\sin(\frac{\pi}{2}x)+2\cos^2(x-1)+(-2,-1,0)x^4=(2,5,8),$$

with the exact solution $x=1$. Using a similar manner which has been described in previous examples, training is started with $x_0=1.5$, η =0.01 and γ = 0.01. Numerical result can be found in Table 4. Similarly, Figures 8 and 9 show the accuracy of the solution $x(t)$.

Table 4. The approximated solutions with error analysis for Example 4.4

t	$x_0(t)$	*e*	*t*	$x_0(t)$	*e*
1	1.3887	640.4183	11	1.0135	0.493675900
2	1.2974	322.2644	12	1.0092	0.228763600
3	1.2234	163.8896	13	1.0062	0.105645100
4	1.1647	83.1223	14	1.0042	0.048672060
5	1.1193	41.67327	15	1.0029	0.022387090
6	1.0850	20.56737	16	1.0019	0.010285530
7	1.0598	9.988083	17	1.0013	0.004721960
8	1.0416	4.780992	18	1.0009	0.002166658
9	1.0287	2.261678	19	1.0006	0.000993581
10	1.0197	1.060221	2	--------	----------

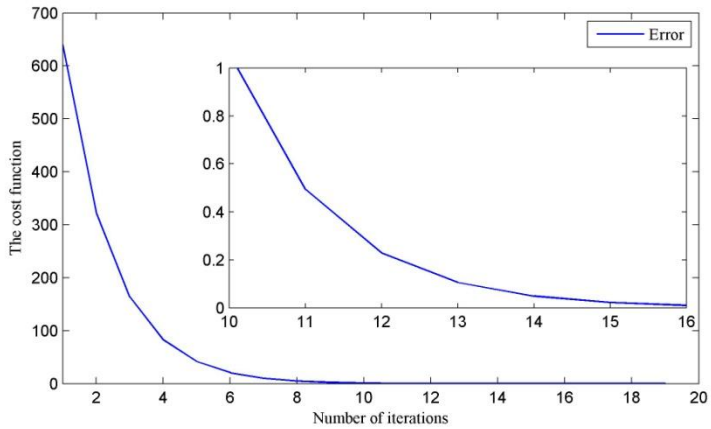

Figure 8. The cost function for Example 4.4 over the number of iterations.

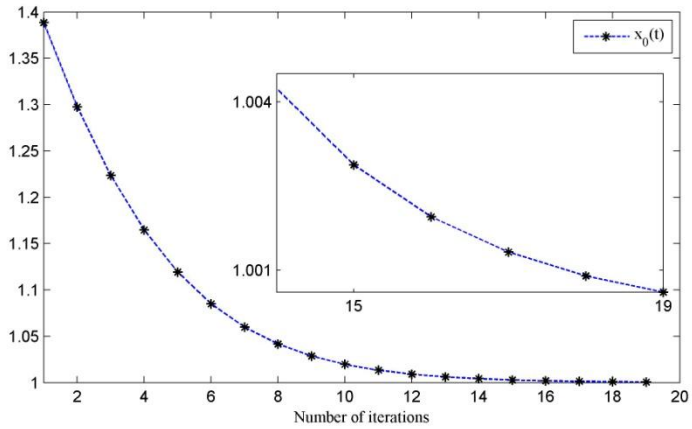

Figure 9. Convergence of the approximated solution for Example 4.4.

5. Conclusions

In this paper, a new architecture of feed-forward neural networks has been proposed to approximate solution of a fuzzy polynomial. Presented FFNN2 in this study is a numerical method for calculating unknown coefficients in the given equation. It is clear that 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 equations. The analyzed examples illustrate the ability and reliability of the present method. The obtained solutions, in comparison with exact solutions admit a remarkable accuracy.

6. References

[1] S. Abbasbandy and M. Alavi, "A method for solving fuzzy linear systems", Iran. J. Fuzzy Syst. 2 (2005) 37-43.
[2] S. Abbasbandy and B. Asady, "Newton's method for solving fuzzy nonlinear equations", Appl. Math. Comput. 159 (2004) 349-356.
[3] S. Abbasbandy and R. Ezzati, "Newton's method for solving a system of fuzzy nonlinear equations", Appl. Math. Comput. 175 (2006) 1189-1199.
[4] S. Abbasbandy and M. Otadi, "Numerical solution of fuzzy polynomials by fuzzy neural network", Appl. Math. Comput. 181 (2006) 1084-1089.
[5] G. Alefeld and J. Herzberger, "Introduction to Interval Computations", Academic Press, New York, 1983.
[6] B. Asady, S. Abbasbandy and M. Alavi, "Fuzzy general linear systems," Appl.Math. Comput. 169 (2005) 34-40.
[7] J.J. Buckley and E. Eslami, "Neural net solutions to fuzzy problems: The quadratic quation", Fuzzy Sets Syst. 86 (1997) 289-298.
[8] J.J. Buckley and Y. Qu, "Solving linear and quadratic fuzzy equations", Fuzzy Sets Syst. 35 (1990) 43-59.
[9] A. Datta, V. Talukdar, A. Konar and L.C. Jain, "A neural network based approach for protein structural class prediction", J. Intell. Fuzzy. Syst. 20 (2009) 61-71.
[10] G. Foggia, T.T. Ha Pham, G.Warkozek and F.Wurtz, "Optimization energy management in buildings with neural networks", Int. J. Appl. Electrom.30 (2009) 237-244.
[11] R. Goetschel and W. Voxman, "Elementary calculus", Fuzzy Sets Syst. 18(1986) 31-43.
[12] Y. Hayashi, J.J. Buckley and E. Czogala, "Fuzzy neural network with fuzzy signals and weights", Int. J. Intell. Syst. 8 (1993) 527-537.

[13] H. Ishibuchi, K.Kwon and H.Tanaka, "A learning of fuzzy neural networks with triangular fuzzy weghts", Fuzzy Sets Syst. 71 (1995) 277-293.
[14] H. Ishibuchi, H. Okada and H. Tanaka, "Fuzzy neural networks with fuzzy weights and fuzzy biases", in: Proc. ICNN 93 (San Francisco), 4 (1993) 1650-1655.
[15] A. Jafarian and R. Jafari, "Approximate solutions of dual fuzzy polynomials by feed-back neural networks", journal of Soft Computation and Applications, (2012). doi:10.5899/2012/jsca-00005
[16] A. Jafarian , S. Measoomy Nia and S. Tavan, "A numerical scheme to solve fuzzy linear volterra integral equations system", Journal of Applied Mathematics, (2012). doi:10.1155/2012/216923.
[17] O. Kaleva, "Fuzyy di erential equations", Fuzzy Sets Syst. 24 (1987) 301-317.
[18] M. Ma, M. Friedman, A. Kandel, "A new fuzzy arithmetic", Fuzzy Sets Syst. 108 (1999) 83-90.
[19] H.T. Nguyen, "A note on the extension principle for fuzzy sets", J. Math. Anal. Appl. 64 (1978) 369-380.
[20] S.K. Oh, W. Pedrycz and S.B. Roh, "Genetically optimized fuzzy polynomial neural networks with fuzzy set-based polynomial neurons", Inform. Sci. 176 (2006) 3490-3519.
[21] A.R. Tahavvor and M. Yaghoubi, "Analysis of natural convection from a column of cold horizontal cylinders using artificial neural network", Appl. Math. Model. 36 (2012) 3176-3188.
[22] C.C. Wang and C.F. Tsai, "Fuzzy processing using polynomial bidirectional hetero associative network", Inform. Sci. 125 (2000) 167-179.
[23] L.A. Zadeh, "The concept of a liguistic variable and its application to approximate reasoning: Parts 1-3", Inform. Sci. 8 (1975) 199-249.
[24] L.A. Zadeh, "Toward a generalized theory of uncertainty (GTU) an outline", Inform. Sci. 172 (2005) 1-40.