Journal of Fuzzy Set Valued Analysis 2017 No.3 (2017) 132-142

Available online at www.ispacs.com/jfsva

Volume 2017, Issue 3, Year 2017 Article ID jfsva-00378, 11 Pages

doi:10.5899/2017/jfsva-00378

Research Article

Application of ANNs approach for solving fully fuzzy polynomials system

R. Novin[1], M. A. Fariborzi Araghi[1*], M. Amirfakhrian[1]

(1) *Department of Mathematics, Central Tehran Branch, Islamic Azad University, Zip code 14676-86831, Tehran, Iran*

Copyright 2017 © R. Novin, M. A. Fariborzi Araghi and M. Amirfakhrian. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract

In processing indecisive or unclear information, the advantages of fuzzy logic and neurocomputing disciplines should be taken into account and combined by fuzzy neural networks. The current research intends to present a fuzzy modeling method using multi-layer fuzzy neural networks for solving a fully fuzzy polynomials system. To clarify the point, it is necessary to inform that a supervised gradient descent-based learning law is employed. The feasibility of the method is examined using computer simulations on a numerical example.

The experimental results obtained from the investigation of the proposed method are valid and delivers very good approximation results.

Keywords: System of fully fuzzy polynomials, Fuzzy feed-forward neural network, Approximate solution, Learning algorithm, Cost function.

1 Introduction

Very often, there have been plenty of problems in some applied fields such as mathematical economics and optimal control theory. In order to find out solutions to solve such problems, the mathematical formulations of physical phenomena that carry fuzzy polynomials are used which found to be useful in the area. Simply said, there already exist numerous methods of solving the problems in the literature. It is productive to narrate the relevant for this work here. Takagi and Sugeno [14] have presented first numerical approach of fuzzy systems. A general model has been proffered for solving a fuzzy linear system using embedding method by Friedman et al. [7]. Further, Dubois and Prade [6] studied theoretical features of a fuzzy linear system . Iterative methods have been utilized to find the solution of fully fuzzy linear system by Dehgan et al. [5]. The solutions to linear and nonlinear fuzzy systems were applied by [1, 2, 4]. In addition, applying fuzzy neural networks (FNNs) meant to be a satisfied tool for finding indirect solutions to fuzzy polynomials system. through determining a cost function for each pair of fuzzy output vector and each of its corresponding fuzzy target vector, Ishibuchi et al. [12] designed a learning algorithm of fuzzy neural networks where the weighs are usually considered as triangular and trapezoidal fuzzy.

Also Hayashi et al. [11] fuzzified the delta learning rule such that Zadeh's extension principle [15] was employed to determine the input-output relation to every unit . Whereas Abbasbandy et al. [3] focused in developing a new learning algorithm to reach a real root of a fuzzy polynomials system in order to illustrate a structure of feed-forward fuzzy neural networks.

*Corresponding author. Email address: m_fariborzi@iauctb.ac.ir; fariborzi.araghi@gmail.com; Tel. +98 2188385773.

It is quite possible to mention that neural networks simulations due to their noticeable efficiency in extracting meaning from vague data, seems to be useful in solving the problems. In this paper, we are eager to construct a new algorithm by using fuzzy neural networks to approximate solution of a fully fuzzy polynomials system. So, a three-layer fuzzy feed-forward neural network (FFNN) which is trainable is used. For the suggested architecture of neural nets, the different power of unknowns are considered as connection weight parameters corresponding to the output layer. First, we aim to adjust the fuzziness of the actual output data to match the desired fuzziness even if the difference between actual and desired modal value is still high. Hence, for the given training patterns, a cost function is defined for the level sets of fuzzy output and target output in which measures the difference between the target outputs and corresponding actual outputs. Therefore, the error function is presented as a function on the fuzzy weight space of the net. By minimizing of the error size, the approximate problem can be achieved. Then, a supervised learning rule based on the gradient descent method is derived for turning the values of the weights into any desired degree of accuracy. Now, the learning problem of the fuzzy net can be considered as a problem of optimization. Eventually, the connection weights are corrected directly after presenting set of training data. This paper is organized as follows. A brief review on the proposed architecture of artificial neural networks and fuzzy numbers is presented in sections 2. Section 3 describes how to find a real solution of the fuzzy system by using FFNN. Finally, an example is illustrated in Section 4. Section 5 concludes the paper.

2 Basic concepts

This section intends to introduce some general and powerful definitions and concepts that will be utilized in the following sections.

2.1 Fuzzy calculus

In this part the most basic used notations in fuzzy calculus are briefly introduced. More detailed information can be found in [8, 13]. We begin by defining the fuzzy number.

Definition 2.1. *A fuzzy number v is a pair $(\underline{v},\overline{v})$ of functions $\underline{v}(r)$ 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) *$\underline{v}(r) \le \overline{v}(r)$: $0 \le r \le 1$.*

For arbitrary fuzzy numbers $u = (\underline{u},\overline{u})$ and $v = (\underline{v},\overline{v})$, we determine addition $(u+v)$ and multiplication by k as follows:

$$\overline{(u+v)}(r) = \overline{u}(r) + \overline{v}(r),$$
$$\underline{(u+v)}(r) = \underline{u}(r) + \underline{v}(r),$$

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

The collection of all fuzzy numbers (as given by definition 1) with addition and multiplication as defined above, is denoted by E^1. For $0 < r \le 1$, a r-level set of fuzzy number u is defined with $[u]^r = \{x|\ u(x) \ge r,\ x \in \mathbb{R}\}$, and for $r = 0$, the support of u is defined as $[u]^0 = \overline{\{x|\ u(x) \ge 0,\ x \in \mathbb{R}\}}$.

Definition 2.2. *Let $u, v \in E^1$. If there exists $w \in E^1$ such that $u = v + w$, then w is called the H-difference of u, v and it is denoted by $u - v$.*

Definition 2.3. *For arbitrary fuzzy numbers $u = (\underline{u},\overline{u})$ and $v = (\underline{v},\overline{v})$ the quantity*

$$D(u,v) = \sup_{0 \le r \le 1} \{\max[|\underline{u}(r) - \underline{v}(r)|, |\overline{u}(r) - \overline{v}(r)|]\},$$

is the distance between u and v.

2.2 An overview on neural networks

An artificial neural network, often just named a neural network, is a novel structure of information processing system that is inspired by biological nervous systems. Using neural network applications, some mathematical problems appear which cannot be conveniently solved with exact formulas. For further readings about the ANNs, see [10, 9].

2.2.*1* *Input-output relation of each unit*

Here, in order to determine an algorithm that will be applied for solving the mentioned fuzzy problem, a brief framework of the proposed architecture of neural networks is to be offered. Consider the two-layer feed-back architecture shown in Figure 1. This network receives input datum of the training set $\{A_{p1},...,A_{pn}\}$ (for $p=1,2$). Also, the activation function of the output units is assumed to be the identity function. In this paper we assume that input vector and the target output are triangular fuzzy numbers and connection weights are crisp numbers. Now, consider set of n training patterns $\{(A_{p1},...,A_{pn});A_{p,0}\}$ (for $p=1,2$), where $A_{p,0}$ refers to the desired network output upon the presentation of the input signals $\{A_{p1},...,A_{pn}\}$. Using the figure, input-output relation of each unit and calculated output Y_p can be written as following:

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

$$o_{pi}=A_{pi},\ i=1,2,...,n. \tag{2.1}$$

Hidden units:

$$O_{pj}=f(net_{pj}),$$
$$net_{pj}=w_j.o_{pj}.$$

Output unit:

$$Y_p=f(Net_p),$$
$$Net_p=\sum_{j=1}^{n}W_j.O_{pj},\ p=1,2;\ j=1,2,...,n, \tag{2.2}$$

where w_j and W_j are weight parameters corresponding to hidden and output network layers, respectively. The relations between input neurons and output neuron in Eqs. (2.1)-(2.2) are defined by the extension principle, as in Hayashi et al. [11].

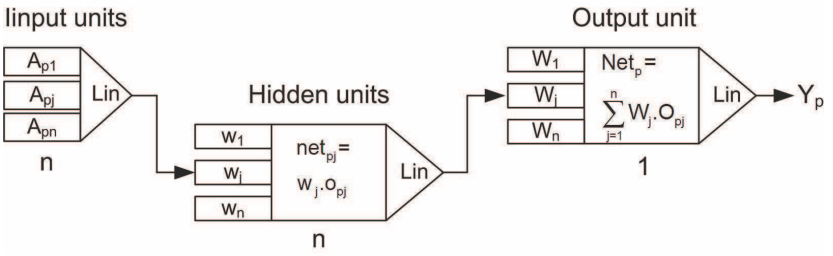


Figure 1: Schematic diagram of the proposed FFNN.

2.2.*2* *Calculation of fuzzy output*

In output layer the fuzzy output neuron is numerically calculated for level sets of fuzzy input signals and fuzzy connection weights. The input-output relations of our fuzzy neural network can be express for the α-level sets as follows:

Input units:

$$[o_{pi}]^{\alpha} = [A_{pi}]^{\alpha},\ i = 1,2,...,n;\ p = 1,2. \tag{2.3}$$

Hidden units:
Let f be an one-to-one activation function. Now we get:

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

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

Output unit:

$$[Y_p]^{\alpha} = f([Net_p]^{\alpha}),$$

$$[Net_p]^{\alpha} = \sum_{j=1}^{n} [W_j.O_{pj}]^{\alpha}. \tag{2.4}$$

For simplify we assume the α-level sets of the fuzzy input A_{pi} is nonnegative, i.e., $0 \leq [A_{pi}]_l^{\alpha} \leq [A_{pi}]_u^{\alpha}$. Considering that the activation function f is identity function, the input-output relations of the neural network can be given for the α-level sets as:

Input units:

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

Hidden units:

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

where

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

and

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

Output unit:

$$[Y_p]^{\alpha} = [[Y_p]_l^{\alpha}, [Y_p]_u^{\alpha}] = [[Net_p]_l^{\alpha}, [Net_p]_u^{\alpha}], \tag{2.5}$$

where

$$[Net_p]^{\alpha} = [[Net_p]_l^{\alpha}, [Net_p]_u^{\alpha}] =$$

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

where $M = \{j|\ [W_j]_l^{\alpha} \geq 0\}$, $C = \{j|\ [W_j]_l^{\alpha} < 0,\ M' = \{j|\ [W_j]_u^{\alpha} \geq 0\}$, $C' = \{j|\ [W_j]_u^{\alpha} < 0\}$, $M \cup C = \{1,...,n\}$ and $M' \cup C' = \{1,...,n\}$.

3 System of fuzzy polynomials

Fuzzy neural nets are globally interconnected feed-forward networks. The processing characters and the neuron's relations of these networks work on fuzzy numbers than real numbers. In this section, we aim to achieve a supervised learning law based on the traditional fuzzy algorithm to the approximate solution of fully fuzzy polynomials system:

$$\begin{cases} A_{11}xy + ... + A_{1n}x^n y^n = A_{10} \\ A_{21}xy + ... + A_{2n}x^n y^n = A_{20} \end{cases}. \tag{3.6}$$

Considering $A=(A_{p,i})_{2\times n}$, $B=(A_{1,0},A_{2,0})^T$ and $X=(xy,...,x^ny^n)$, Eq. (3.6) can be transformed to the below form:

$$AX=B, \tag{3.7}$$

where A is a matrix of fuzzy number entries, X and B are fuzzy number vectors. As seen, an architecture of FFNN solution to Eq. (3.7) is illustrated in Fig. 1. The straightforward and flexible fuzzy neural network architecture is of modeling scheme. Considering the following procedure, the output of the proposed neural architecture with offered conditions, derives a representation for the given fuzzy problem.

3.1 Cost function

Consider the three-layer fuzzy feed-forward architecture shown in Fig. 1. Let A_{p0} be fuzzy target output corresponding to the fuzzy input vector $A_p=(A_{p1},...,A_{pn})$, $w_i=x^i$ and $W_j=y^j$ (for $i,j=1,...,n$). So, we have:

$$[[W_j]_l^\alpha,[W_j]_u^\alpha]=$$

$$\begin{cases} [([y_0]_l^\alpha)^j,([y_0]_u^\alpha)^j] & ,0\le[y_0]_l^\alpha\le[y_0]_u^\alpha \\ \begin{cases} [([y_0]_l^\alpha)^j,([y_0]_u^\alpha)^j] & ,j \text{ is an odd integer} \\ [([y_0]_u^\alpha)^j,([y_0]_l^\alpha)^j] & ,j \text{ is an even integer} \end{cases} & ,[y_0]_l^\alpha\le[y_0]_u^\alpha<0 \end{cases},$$

$$[[w_i]_l^\alpha,[w_i]_u^\alpha]=$$

$$\begin{cases} [([x_0]_l^\alpha)^i,([x_0]_u^\alpha)^i] & ,0\le[x_0]_l^\alpha\le[x_0]_u^\alpha \\ \begin{cases} [([x_0]_l^\alpha)^i,([x_0]_u^\alpha)^i] & ,i \text{ is an odd integer} \\ [([x_0]_u^\alpha)^i,([x_0]_l^\alpha)^i] & ,i \text{ is an even integer} \end{cases} & ,[x_0]_l^\alpha\le[x_0]_u^\alpha<0 \end{cases}.$$

The aim is to define and minimize a cost function to the proposed network, which compute the error in the actual output data on the set of training data. Therefore this function is refferred to a function on the weight parameters space of the network. So, the cost function of a fuzzy network can be provided for α-level sets of fuzzy output Y_p and its corresponding target output A_{p0} by:

$$e_p^\alpha=e_{pl}^\alpha+e_{pu}^\alpha, \tag{3.8}$$

where

$$e_{pl}^\alpha=\alpha.\frac{([A_{p0}]_l^\alpha-[Y_p]_l^\alpha)^2}{2}, \tag{3.9}$$

$$e_{pu}^\alpha=\alpha.\frac{([A_{p0}]_u^\alpha-[Y_p]_u^\alpha)^2}{2}. \tag{3.10}$$

In the relation (3.8), e_{pl}^α and e_{pu}^α denote 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. Now the cost function for the training pattern $\{A_P;A_{p0}\}$ is determined as [12]:

$$e_p=\sum_\alpha e_p^\alpha. \tag{3.11}$$

3.2 Learning algorithm of the FFNN

In artificial neural networks, learning is of great importance in creating network parameters which neccessary drives the output error to zero. Here the purpose is to apply the supervised gradient descent-based learning procedure that is a natural generalization of the delta learning rule in which it tends is to minimize the error function on the weight space by finding the weight parameters. Let the trapezoidal fuzzy quantities $a_1=(x_0^1,x_0^2,x_0^3,x_0^4)$ and $a_2=(y_0^1,y_0^2,y_0^3,y_0^4)$ are initialized at small random values for variables x and y, respectively.
In order to set the mentioned learning rule to the above quantities, consider the following. Consider this learning rule:

$$a_j^r(t+1)=a_j^r(t)+\Delta a_j^r(t),\quad r=1,2,3,4, \tag{3.12}$$

$$\Delta a_j^r(t) = -\eta . \frac{\partial e_p(\alpha)}{\partial a_j^r} + \gamma . \Delta a_j^r(t-1),\ p = 1,2, \tag{3.13}$$

where t is the number of adjustments, η and α are the small constant learning rate and the momentum term constant in which normally chosen between 0 and 1, respectively. The network parameters are updated to deduct the error and then the network output converges for each given input to the desired output. To do this, the new value for each input is found by taking the current input and adding an amount that is proportional to the slope of training. Now, the partial derivative $\frac{\partial e_p^\alpha}{\partial y_0}$ is evaluated at the current fuzzy weight values. Having the chain rule for differentiation, one may illustrate the present partial derivative as:

$$\frac{\partial e_p^\alpha}{\partial y_0} = \frac{\partial e_{pl}^\alpha}{\partial y_0} + \frac{\partial e_{pu}^\alpha}{\partial y_0}. \tag{3.14}$$

Certainly, there could be a problem in calculating the $\frac{\partial e_{pl}^\alpha}{\partial y_0}$ and $\frac{\partial e_{pu}^\alpha}{\partial y_0}$ derivatives. however, calculating both of them can be done in the same manner, we illustrate the calculation of $\frac{\partial e_{pl}^\alpha}{\partial y_0}$. So, we have:

$$\frac{\partial e_p^l(\alpha)}{\partial a_j^r} = \frac{\partial e_p^l(\alpha)}{\partial [Y_p]_l^\alpha} . \frac{\partial [Y_p]_l^\alpha}{\partial [Net_p]_l^\alpha} . \frac{\partial [Net_p]_l^\alpha}{\partial a_j^r} = -([A_{p0}]_\alpha^l - [Y_p]_\alpha^l) . \frac{\partial [Net_p]_l^\alpha}{\partial a_j^r},\ j = 1,2,$$

where

$$\frac{\partial [Net_p]_l^\alpha}{\partial a_2^r} = \frac{\partial [Net_p]_l^\alpha}{\partial [W_j]_l^\alpha} . \frac{\partial [W_j]_l^\alpha}{\partial a_2^r} = \begin{cases} [O_{pj}]_l^\alpha . \frac{\partial [W_j]_l^\alpha}{\partial a_2^r} & , j \varepsilon M \\ [O_{pj}]_u^\alpha . \frac{\partial [W_j]_l^\alpha}{\partial a_2^r} & , j \varepsilon C \end{cases},$$

$$\frac{\partial [W_j]_l^\alpha}{\partial a_2^r} = \begin{cases} j.([y_0]_l^\alpha)^{j-1} . \begin{cases} 1-\alpha & , r=1 \\ \alpha & , r=2 \\ 0 & , r=3 \\ 0 & , r=4 \end{cases} & , [y_0]_l^\alpha \geq 0 \\ \begin{cases} j.([y_0]_l^\alpha)^{j-1} . \begin{cases} 1-\alpha & , r=1 \\ \alpha & , r=2 \\ 0 & , r=3 \\ 0 & , r=4 \end{cases} & , \ j \text{ is odd} \\ j.([y_0]_u^\alpha)^{j-1} . \begin{cases} 0 & , r=1 \\ 0 & , r=2 \\ \alpha & , r=3 \\ 1-\alpha & , r=4 \end{cases} & , \ j \text{ is even} \end{cases} & , [y_0]_u^\alpha \leq 0 \end{cases}.$$

Using a similar procedure as an outline which mentioned above, we have the correspondingly corollary for partial derivative $\frac{\partial [Net_p]_l^\alpha}{\partial a_1^r}$, in which we refrain to go through proven details. In addition, the proposed neural network parameter a_1^r is updated in which it owns a simplest extension of the rule that has been given above. So, the details will be left for the readers to look for. It should be mentioned that, the derivatives of $[Net_p]_l^\alpha$, with respect to variable a_1^r is:

$$\frac{\partial [Net_p]_l^\alpha}{\partial a_1^r} = \begin{cases} \frac{\partial [Net_p]_l^\alpha}{\partial [O_{pj}]_l^\alpha} . \frac{\partial [O_{pj}]_l^\alpha}{\partial a_1^r} & , j \varepsilon M \\ \frac{\partial [Net_p]_l^\alpha}{\partial [O_{pj}]_u^\alpha} . \frac{\partial [O_{pj}]_u^\alpha}{\partial a_1^r} & , j \varepsilon C \end{cases} = \begin{cases} [W_j]_l^\alpha . [A_{pj}]_l^\alpha . \frac{\partial [w_j]_l^\alpha}{\partial a_1^r} & , j \varepsilon M \\ [W_j]_l^\alpha . [A_{pj}]_u^\alpha . \frac{\partial [w_j]_u^\alpha}{\partial a_1^r} & , j \varepsilon C \end{cases},$$

where

$$\frac{\partial [w_j]_l^\alpha}{\partial a_1^r} = \begin{cases} j([x_0]_l^\alpha)^{j-1} \begin{cases} 1-\alpha & ,r=1 \\ \alpha & ,r=2 \\ 0 & ,r=3 \\ 0 & ,r=4 \end{cases} & ,[x_0]_l^\alpha \geq 0 \\ \begin{cases} j([x_0]_l^\alpha)^{j-1} \begin{cases} 1-\alpha & ,r=1 \\ \alpha & ,r=2 \\ 0 & ,r=3 \\ 0 & ,r=4 \end{cases} & ,j \text{ is odd} \\ j([x_0]_u^\alpha)^{j-1} \begin{cases} 0 & ,r=1 \\ 0 & ,r=2 \\ \alpha & ,r=3 \\ 1-\alpha & ,r=4 \end{cases} & ,j \text{ is even} \end{cases} & ,[x_0]_u^\alpha < 0 \end{cases}.$$

Considering the input-output pair $\{A_p;A_{p0}\}$ where $A_p = (A_{p1},...,A_{pn})$ and different arbitrary values for α-level sets $(i.e.\ \alpha_1,\alpha_2,...,\alpha_m)$, learning process of suggested algorithm consists of the following five stages:

Learning process
Stage 1: Set the learning rate, momentum constant and *Emax* to minimum affirmative values. Then, initialize all initial fuzzy connection weights.
Stage 2: Set $t := 0$ where t is number of learning iterations and also the running error E is set to 0.
Stage 3: Set $t := t+1$. Repeat following procedure for different values of p,q:

i) Forward calculation: Choose the input pattern A_p from the training set and distribute it through the network. Then approximate actual fuzzy output Y_p based on the current weights space.

ii) Back-propagation: Update all weights according to the Eq. (3.12) for the hidden and output layers.

Stage 4: Accumulative cycle error is computed by adding the present error to E.
Stage 5: The training cycle is completed. For $E < Emax$, cease the training session. If $E > Emax$ then E is set to 0 and a new training cycle is initiated by turning back to *Stage 3*.

It is, now, convenient to note that, back-propagation is found to be extremely sensitive to initial conditions. Subsequently, if the training cycle is complex, it may not provide a possible solution. In such case, the learning parameters must be measured with new values where the convergence speed of the recommended process is highly related to the learning rate and momentum constant parameters.

4 Numerical example

Trying to find the approximate fuzzy solution of the present example, the proposed method is selected to do so. In the following simulations, we consider the specifications as follows:

1) Learning rate $\eta = 0.01$,

2) Momentum constant $\alpha = 0.01$,

3) Stopping condition $Emax = 10^{-4}$.

Example 4.1. *Let the following system of fully fuzzy polynomials:*

$$\begin{cases} (2,3,4,5)xy & + & (1,3,6)x^2y^2 & = & (8,126,340,5550) \\ (1,3,4)xy & + & (3,5,6,9)x^2y^2 & = & (14,198,630,8220) \end{cases}.$$

Where the exact solution is $x = (1,2,2,5)$ *and* $y = (2,3,5,6)$. *A point should be clarified that the present problem is solved using neural network architecture which is provided in this paper, considering the network parameters* a_1 *and* a_2 *are quantified with* $(0.1,0.2,0.3,0.4)$ *and* $(0.5,0.6,0.7,0.8)$, *respectively. The cost function on the number of iterations is illustrated in Figure 2. More importantly, the more the iterations on the increase, the more the cost function would lead to zero. Further, collected numerical results on the various number of iterations, can be seen in Table 1.*
Also, Figure 3 shows the convergence of the approximate solutions obtained from the proposed algorithm.

Table 1: The approximated solution with error analysis for Example 4.1.

t	$a_1(t)$	$a_2(t)$	$Error$
1	(0.1186, 0.2007, 0.3820, 0.4892)	(0.5825, 0.6786, 0.7904, 0.9112)	15.5668
2	(0.1834, 0.5618, 0.4710, 1.2321)	(0.7435, 0.9086, 1.4630, 1.7021)	10.7425
3	(0.2186, 0.7980, 0.8906, 1.8867)	(0.8360, 1.1527, 1.8196, 2.2850)	8.86985
4	(0.2537, 1.1318, 1.1038, 2.3705)	(0.9117, 1.3804, 2.4706, 2.7098)	7.39713
5	(0.2807, 1.3572, 1.4530, 2.9001)	(1.2608, 1.7411, 3.0120, 3.3704)	6.21443
⋮	⋮	⋮	⋮
166	(0.9980, 1.9975, 1.9984, 4.9961)	(1.9971, 2.9978, 4.9973, 5.9962)	0.00169
167	(0.9985, 1.9981, 1.9987, 4.9966)	(1.9976, 2.9980, 4.9976, 5.9968)	0.00135
168	(0.9988, 1.9988, 1.9990, 4.9972)	(1.9980, 2.9984, 4.9981, 5.9973)	0.00121
169	(0.9990, 1.9991, 1.9993, 4.9977)	(1.9986, 2.9990, 4.9987, 5.9980)	0.00108
170	(0.9993, 1.9995, 1.9998, 4.9980)	(1.9997, 2.9998, 4.9990, 5.9987)	0.00094

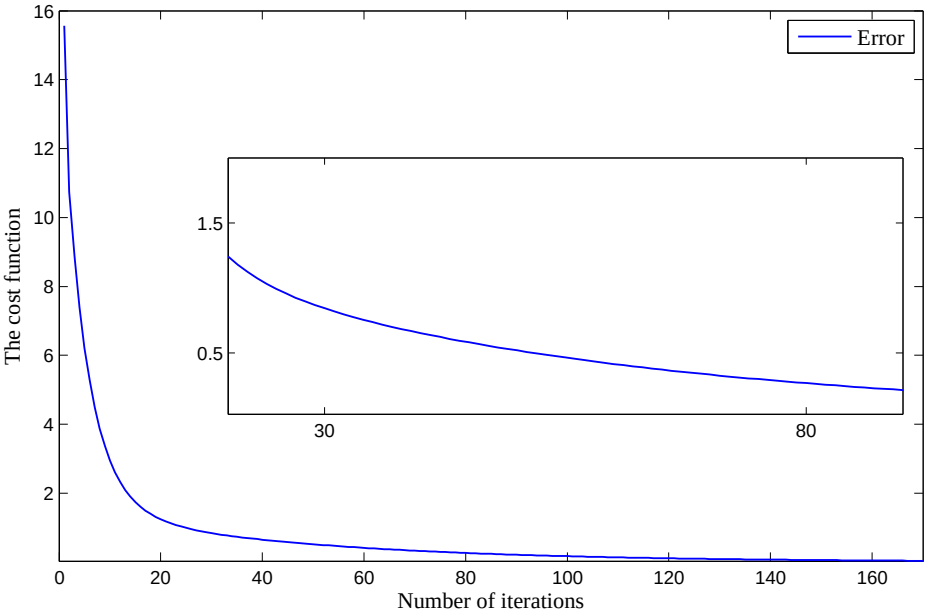
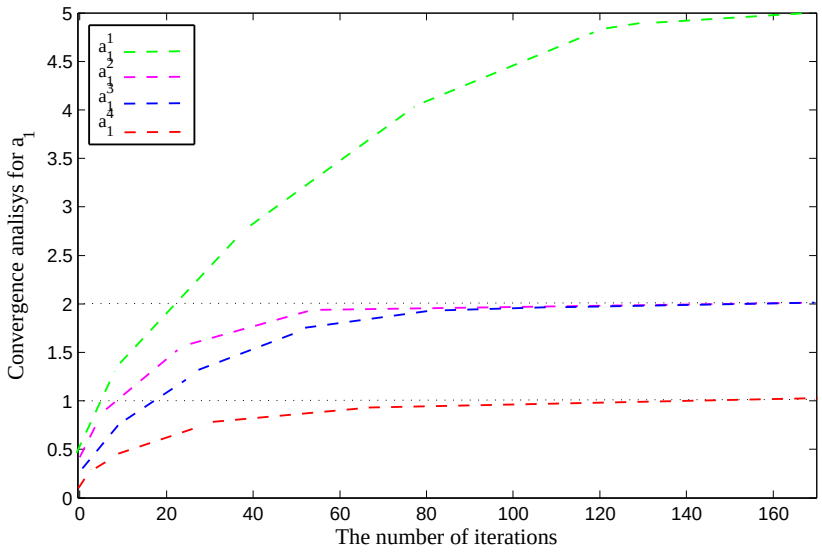
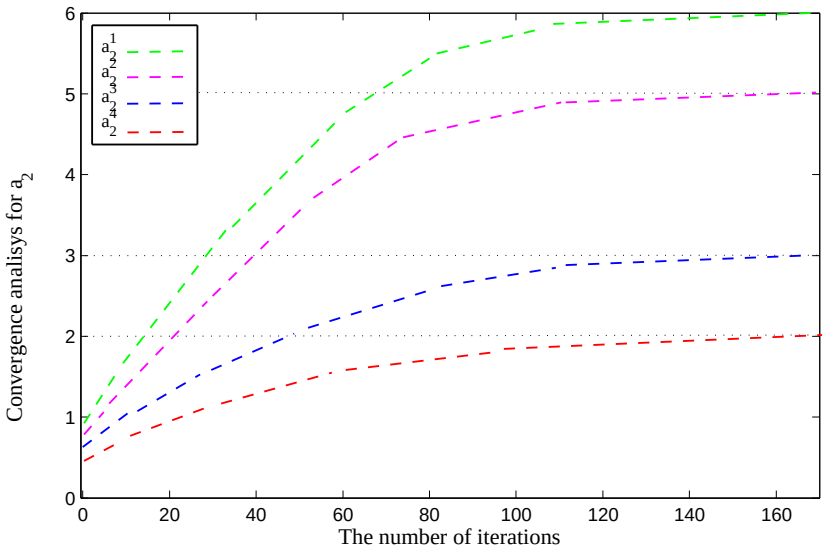


Figure 2: The cost function for Example 4.1 over the number of iterations.



a *the solution* a_1



b *the solution* a_2

Figure 3: Convergence of the approximate solutions for Example 4.1.

5 Conclusion

In short, artificial neural networks can deal with the issues outside the range of conventional processors due to the fact that they are not conventionally programmed yet. A worldwide iterative approach using ANNs, which is convenient for approximating a fuzzy polynomials system, is discussed in present paper. The proposed multi-layer feed forward architecture selects a computationally efficient training method that is based on a supervised gradient

descent-based learning rule to approximate solutions of the fuzzy problems. To check the validity of the method, one numerical example was provided. Obtained results clarify that the proposed technique creates a powerful tool which is demanded for solving the fuzzy systems. An important point should be made here that if the present method developed with a claim that the number of iterations are chosen large enough, the method is said to have the best approximate solution. Having this knowledge, any attempt to establish the approximate solution of any kinds of fuzzy systems can be quite convenient. And last, the analyzed examples really did determine the extent of the ability and high validity of the proposed method without any ambiguity.

References

[1] S. Abbasbandy, M. Alavi, A method for solving fuzzy linear systems, Iran. J. Fuzzy Syst, 2 (2005) 37-43.
http://ijfs.usb.ac.ir/article_481_0.html

[2] S. Abbasbandy, R. Ezzati, Newton's method for solving a system of fuzzy nonlinear equations, Appl. Math. Comput, 175 (2006) 1189-1199.
https://doi.org/10.1016/j.amc.2005.08.021

[3] S. Abbasbandy, M. Otadi, M. Mosleh, Numerical solution of a system of fuzzy polynomials by fuzzy neural network, Informa. Sci, 178 (2008) 1948-1960.
https://doi.org/10.1016/j.ins.2007.11.026

[4] B. Asady, S. Abbasbandy, M. Alavi, Fuzzy general linear systems, Appl. Math. Comput, 169 (2005) 34-40.
https://doi.org/10.1016/j.amc.2004.10.042

[5] M. Dehgan, B. Hashemi, M. Ghatee, Solution of the fully fuzzy linear systems using iterative techniques, Chaos Solitons Fract, 34 (2)(2007) 316-336.
https://doi.org/10.1016/j.chaos.2006.03.085

[6] D. Dubois, H. Prade, Systems of linear fuzzy constraints, Fuzzy Sets Syst, 3 (1980) 37-48.
https://doi.org/10.1016/0165-0114(80)90004-4

[7] M. Friedman, M. Ming, A. Kandel, Fuzzy linear systems, Fuzzy Sets Syst, 96 (1998) 201-209.
https://doi.org/10.1016/S0165-0114(96)00270-9

[8] R. Goetschel, W. Voxman, Elementary calculus, Fuzzy Sets Syst, 18 (1986) 31-43.
https://doi.org/10.1016/0165-0114(86)90026-6

[9] N. K. Goyal, A. Kumar, S. Trivedi, U. S. Dwivedi, T. N. Singh, P. B. Singh, A comparative study of artificial neural network and multivariate regression analysis to analyze optimum renal stone fragmentation by extracorporeal shock wave lithotripsy, Saudi Journal of Kidney Diseases and Transplantation, 21 (6) (2010) 1073-1076.
https://www.ncbi.nlm.nih.gov/pubmed/21060176

[10] D. Graupe, Principles of artificial neural networks (2nd Edition), World Scientific Publishing, (2007).
https://doi.org/10.1142/6429

[11] Y. Hayashi, J. J. Buckley, E. Czogala, Fuzzy neural network with fuzzy signals and weights, Int. J. Intell. Syst, 8 (1993) 527-537.
https://doi.org/10.1002/int.4550080405

[12] H. Ishibuchi, K. Kwon, H. Tanaka, A learning of fuzzy neural networks with triangular fuzzy weghts, Fuzzy Sets Syst, 71 (1995) 277-293.
https://doi.org/10.1016/0165-0114(94)00281-B

[13] H. T. Nguyen, A note on the extension principle for fuzzy sets, J. Math. Anal. Appl, 64 (1978) 369-380.
https://doi.org/10.1016/0022-247X(78)90045-8

[14] T. Takagi, M. Sugeno, Structure identifical of systems and its application to modelling and control, IEEE Trans.Systems Man Cybern, 15 (1985) 116-132.
https://doi.org/10.1109/TSMC.1985.6313399

[15] L. A. Zadeh, The concept of a linguistic variable and its application to approximate reasoning, Part 1 Inform. Sci. 8, 199-249; Part 2. Inform. Sci. 8, 301-353; Part 3. Inform. Sci. 9, 43-80.
https://doi.org/10.1016/0020-0255(75)90036-5