Advances in Fuzzy Mathematics.
ISSN 0973-533X Volume 8, Number 1 (2013), pp. 25-33

http://www.ripublication.com/afm.htm

New Method for Solving Fuzzy Polynomials

A. Jafarian
Department of Mathematics, Urmia Branch,
Islamic Azad University, Urmia, Iran
E-mail: jafarian5594@yahoo.com

R. Jafari
Department of Mathematics,
Science and Research Branch,
Islamic Azad University, Arak, Iran
E-mail: jafari3339@yahoo.com

Abstract

This paper mainly intends to offer a new method for finding a fuzzy solution of a fully fuzzy polynomial (FFP) with degree one that supposedly has a fuzzy solution. For this scope, an approach based on fuzzy neural network (FNN) is presented. The proposed FNN is a three layer feed-back neural network, where it can get a fuzzy input signal and calculates its corresponding fuzzy output. In order to find the approximate solution of these polynomials, first a cost function for the level sets of fuzzy output and fuzzy target is defined. Then for adjusting the three parameters of triangular fuzzy weight a learning algorithm which is based on the gradient descent method is introduced. Finally, we illustrate our approach by some numerical examples.

AMS subject classification:
Keywords: Fully fuzzy polynomials, Fuzzy feed-back neural networks, Learning algorithm, Cost function, Fuzzy solution.

1. Introduction

The fuzzy polynomial approach is very useful for solving many problems in several applied fields like physical applications, potential theory, electrostatics, mathematical problems and radiative heat transfer problems. Since the fuzzy polynomials are usually difficult to solve analytically so, several numerical approaches for finding the solution of

these polynomials have been reported. One approach to indirect solution is using FNNs. Ishibuchi et al. [7] proposed a learning algorithm of FNNs with triangular fuzzy weights and Hayashi et al. [6] also fuzzified the delta rule. Linear and nonlinear fuzzy equations have been solved in [3, 4]. Also Abbasbandy [1] has proposed an architecture of feed-forward fuzzy neural network for finding crisp solution to fuzzy polynomials. Jafarian et al. [9] solved fuzzy polynomials by using the neural networks with a new learning algorithm. We refer the reader to [10, 11] for more information on fuzzy polynomials.

In this paper, we are interested in finding solution to a FFP with degree one. We wish to find the answer to this question: How is the fuzzified feed-back neural network going to solve the FFP? For answering to this question, we first propose an architecture of fuzzified feed-back neural network with fuzzy weight, fuzzy input and fuzzy target. The input-output relation of each unit is defined by the extension principle of Zadeh [12]. The output from the FNN, which is a fuzzy number, is numerically calculated by interval arithmetic [2] for the level sets (i.e., α-cuts) of fuzzy input and fuzzy weight. Next we define a cost function for the level sets of fuzzy output and fuzzy target. Then, a fuzzy learning algorithm is derived from the cost function to find a fuzzy root of FFP. The proposed algorithm illustrated by solving some examples in the last section.

2. Foundations

In the last years various approach for solving fuzzy equations have been proposed. In [4], by using the classical methods which was based on the extension principle, the investigation for finding solutions to linear and quadratic equations when the coefficients were real or complex fuzzy numbers have been shown that, too often these equations do not have a solution. This result have been caused to investigate other solutions to fuzzy equations. In [12] it has been shown that the fuzzy quadratic equation, where the coefficients are all real fuzzy numbers, always has a solution as a real or complex fuzzy number. Since the particular case of fuzzy equation is fuzzy polynomial so we describe about it in continuance.

Definition 2.1. Artificial neural systems or neural networks are physical cellular systems which can acquire, store and utilize experiential knowledge.

Neural networks can be considered as simplified mathematical models of brain and they function as parallel distributed computing networks.

The following theorem illustrates the convergence properties of the neural networks.

Theorem 2.2. If the presented fuzzy problem has solutions then the perceptron learning algorithm will find one of them.

Proof. See [5]. ■

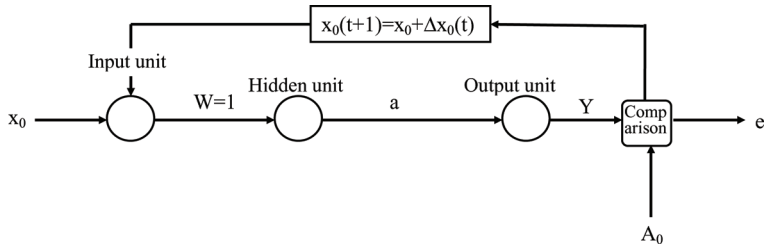

Figure 1: Feed-back fuzzy neural network to solve fuzzy polynomials.

2.1. Fully fuzzy polynomials

In this part, we want to find fuzzy solution of

$$ax = A_0, \tag{1}$$

when $a, x, A_0 \in E^1$. For getting an approximate solution, an architecture of FNN3 (fuzzy neural network with fuzzy input signal and fuzzy weight and fuzzy target) equivalent to Eq. (1) has been given in Fig. 1. Since the transfer function is identity, so the input neuron make no change in its input, therefore the output neuron is $Y = ax$.

2.2. Calculation of fuzzy output

Let us fuzzify a three-layer FNN3 with one input unit, one neuron in hidden layer and one output unit. In order to derive a learning rule, we restrict fuzzy weight within triangular fuzzy number while we can use any type of fuzzy numbers for fuzzy input and fuzzy target. The input-output relation of each unit can be written for the α-level sets as follows when the α-level sets of the fuzzy input a is nonnegative, i.e., $0 \leq [a]^l_\alpha \leq [a]^u_\alpha$:
Input unit:

$$[o]_\alpha = [[o]^l_\alpha, [o]^u_\alpha] = [[x_0]^l_\alpha, [x_0]^u_\alpha].$$

Hidden units:

$$[O]_\alpha = [[O]^l_\alpha, [O]^u_\alpha] = [f([net]^l_\alpha), f([net]^u_\alpha)].$$

Output unit:

$$[Y]_\alpha = [[Y]^l_\alpha, [Y]^u_\alpha] = [f([Net]^l_\alpha), f([Net]^u_\alpha)], \tag{2}$$

If $0 \leq [O]^l_\alpha$, then $[Net]^l_\alpha = [O]^l_\alpha.[a]^l_\alpha$.
If $[O]^l_\alpha < 0$, then $[Net]^l_\alpha = [O]^l_\alpha.[a]^u_\alpha$.
If $0 \leq [O]^u_\alpha$, then $[Net]^u_\alpha = [O]^u_\alpha.[a]^u_\alpha$.
If $[O]^u_\alpha < 0$, then $[Net]^u_\alpha = [O]^u_\alpha.[a]^l_\alpha$.

2.3. Cost function

Let the α-level sets of the fuzzy target output A_0 are denoted by

$$[A_0]_\alpha = [[A_0]^l_\alpha \ , \ [A_0]^u_\alpha], \quad \alpha \in [0,1],$$

where $[A_0]^l_\alpha$ denotes the left-hand side and $[A_0]^u_\alpha$ denotes the right-hand side of the α-level sets of the desired output. A cost function to be minimized is defined for each α-level set as follows:

$$e_\alpha = e^l_\alpha + e^u_\alpha, \tag{3}$$

where

$$e^l_\alpha = \alpha.\frac{([A_0]^l_\alpha - [Y]^l_\alpha)^2}{2}, \tag{4}$$

$$e^u_\alpha = \alpha.\frac{([A_0]^u_\alpha - [Y]^u_\alpha)^2}{2}. \tag{5}$$

In the cost function (3), 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. Then the total error of the given neural network is obtained as:

$$e = \sum_\alpha e_\alpha. \tag{6}$$

Theoretically this cost function satisfies the following equation if we use infinite number of α-level sets in Eq. (6)

$$e \longrightarrow 0 \Longleftrightarrow [Y]^\alpha \longrightarrow [A_0]^\alpha.$$

2.3.1 Learning of fuzzy neural network

Let us denote the triangular fuzzy weight x_0 by its three parameters as $x_0 = (x^1_0, x^2_0, x^3_0)$. Our main aim is adjusting the parameter x_0 by using the learning algorithm which be introduced in below. For fuzzy parameter x_0 adjustment rule can be written as follows [7]:

$$x^p_0(t+1) = x^p_0(t) + \Delta x^p_0(t), \quad p = 1,2,3,$$

$$\Delta x^p_0(t) = -\eta.\frac{\partial e_\alpha}{\partial x^p_0} + \gamma.\Delta x^p_0(t-1), \tag{7}$$

where t is the number of adjustments, η is the learning rate and γ is the momentum term constant. We calculate $\frac{\partial e_\alpha}{\partial x_0^p}$ as follows:

$$\frac{\partial e_\alpha}{\partial x_0^p} = \frac{\partial e_\alpha^l}{\partial x_0^p} + \frac{\partial e_\alpha^u}{\partial x_0^p}, \tag{8}$$

where

$$\frac{\partial e_\alpha^l}{\partial x_0^p} = \frac{\partial e_\alpha^l}{\partial [Y]_\alpha^l}.\frac{\partial [Y]_\alpha^l}{\partial [Net]_\alpha^l}.\frac{\partial [Net]_\alpha^l}{\partial [net]_\alpha^l}.\frac{\partial [net]_\alpha^l}{\partial [x_0]_\alpha^l}.\frac{\partial [x_0]_\alpha^l}{\partial x_0^p},$$

$$\frac{\partial e_\alpha^u}{\partial x_0^p} = \frac{\partial e_\alpha^u}{\partial [Y]_\alpha^u}.\frac{\partial [Y]_\alpha^u}{\partial [Net]_\alpha^u}.\frac{\partial [Net]_\alpha^u}{\partial [net]_\alpha^u}.\frac{\partial [net]_\alpha^u}{\partial [x_0]_\alpha^u}.\frac{\partial [x_0]_\alpha^u}{\partial x_0^p}.$$

If $0 \leq [O]_\alpha^l \leq [O]_\alpha^u$, then

$$\frac{\partial e_\alpha}{\partial x_0^1} = -\alpha.([A_0]_\alpha^l - [Y]_\alpha^l).[a]_\alpha^l.(1-\alpha),$$

$$\frac{\partial e_\alpha}{\partial x_0^2} = -\alpha^2.([A_0]_\alpha^l - [Y]_\alpha^l).[a]_\alpha^l - \alpha^2.([A_0]_\alpha^u - [Y]_\alpha^u).[a]_\alpha^u,$$

$$\frac{\partial e_\alpha}{\partial x_0^3} = -\alpha.([A_0]_\alpha^u - [Y]_\alpha^u).[a]_\alpha^u.(1-\alpha).$$

If $[O]_\alpha^l < [O]_\alpha^u < 0$, then

$$\frac{\partial e_\alpha}{\partial x_0^1} = -\alpha.([A_0]_\alpha^l - [Y]_\alpha^l).[a]_\alpha^u.(1-\alpha),$$

$$\frac{\partial e_\alpha}{\partial x_0^2} = -\alpha^2.([A_0]_\alpha^l - [Y]_\alpha^l).[a]_\alpha^u - \alpha^2.([A_0]_\alpha^u - [Y]_\alpha^u).[a]_\alpha^l,$$

$$\frac{\partial e_\alpha}{\partial x_0^3} = -\alpha.([A_0]_\alpha^u - [Y]_\alpha^u).[a]_\alpha^l.(1-\alpha).$$

If $[O]_\alpha^l < 0 \leq [O]_\alpha^u$, then

$$\frac{\partial e_\alpha}{\partial x_0^1} = -\alpha.([A_0]_\alpha^l - [Y]_\alpha^l).[a]_\alpha^u.(1-\alpha),$$

$$\frac{\partial e_\alpha}{\partial x_0^2} = -\alpha^2.([A_0]_\alpha^l - [Y]_\alpha^l).[a]_\alpha^u - \alpha^2.([A_0]_\alpha^u - [Y]_\alpha^u).[a]_\alpha^u,$$

$$\frac{\partial e_\alpha}{\partial x_0^3} = -\alpha.([A_0]_\alpha^u - [Y]_\alpha^u).[a]_\alpha^u.(1-\alpha).$$

Now we can update the connection weight w as follows:

$$w = x_0. \tag{9}$$

Learning algorithm
Step 1: $\eta > 0$, $\gamma > 0$ and $Emax > 0$ are chosen. Then the fuzzy number x_0 is initialized at a small 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: Let $t := t + 1$. Repeat *Step 5* for $\alpha = \alpha_1, \ldots, \alpha_m$.
Step 4: The following procedures are calculated:

[i] Forward calculation: Calculate the α-level set of the fuzzy output Y by presenting the α-level set of the fuzzy coefficient a.

[ii] Back-propagation: Adjust fuzzy parameter x_0 by using the cost function for the α-level sets of the fuzzy output Y and the target output A_0 then update the connection weight by using the Eq.(9).

Step 5: Update the corresponding connection weight to the hidden layer as $[w]_\alpha = [x_0]_\alpha$.
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*.

3. Numerical examples

In the following, we study some examples of fuzzy polynomials.

Example 3.1. Let fuzzy equation

$$(4,5,6,7)x = A_0,$$

where

$$(\underline{A}_0(r), \overline{A}_0(r)) = (r^2 + 6r + 8\ ,\ r^2 - 11r + 28),\ \ 0 \le r \le 1.$$

and the exact solution is $x = (2,3,4)$. Before starting calculations, we assumed that $x_0 = (4,5,6)$, $\eta = 0.001$ and $\gamma = 0.001$. Numerical result can be found in Table 1.

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

t	$x_0(t)$			e
1	(3.8275	4.8548	5.9022)	344.852512
2	(3.2454	4.4341	5.4227)	141.614621
3	(2.8215	3.9923	4.9632)	59.4807712
4	(2.5353	3.6008	4.5663)	26.0349901
5	(2.3138	3.2412	4.2987)	12.2062001
⋮		⋮		⋮
40	(2.0049	3.0011	4.0025)	0.00196916
41	(2.0044	3.0010	4.0023)	0.00158992
42	(2.0040	3.0009	4.0020)	0.00128372
43	(2.0036	3.0008	4.0018)	0.00103648
44	(2.0032	3.0007	4.0015)	0.00083687

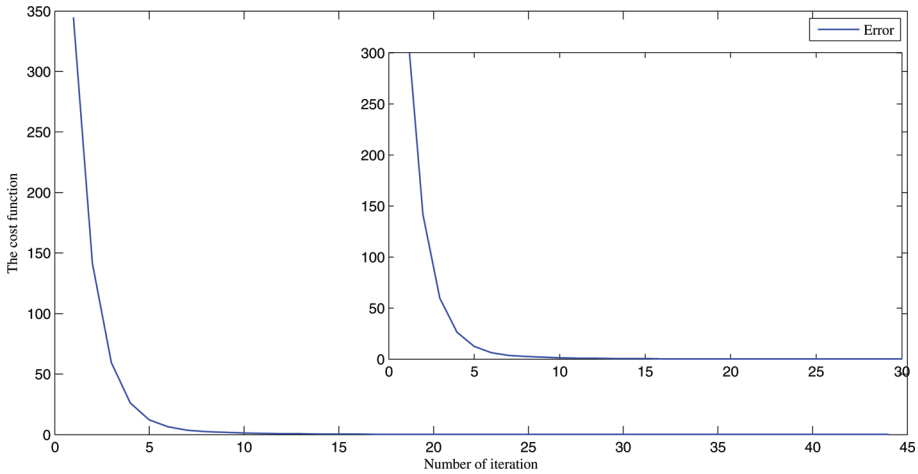

Figure 2: The cost function for Example 1 on the number of iterations.

Fig. 2 shows the accuracy of the solution $x_0(t)$ where t is the number of iterations, its noticeable that by increasing the iteration the cost function goes to zero.

Example 3.2. Consider the following fuzzy equation problem:

$$(2,3,4)x = A_0,$$

where

$$(\underline{A}_0(r), \overline{A}_0(r)) = (-r^2 + 7r - 12\,,\ -r^2 - 3r - 2),\ \ 0 \le r \le 1.$$

The exact solution is $x = (-3,-2,-1)$. This problem is solved with the help of fuzzy neural network as described in this paper. Let $x_0 = (-5,-4,-3)$, $\eta = 0.001$

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

t	$x_0(t)$			e
1	(-4.8068	-3.8530	-2.9093)	146.853251
2	(-4.3466	-3.4298	-2.5130)	103.323251
3	(-3.9138	-3.0263	-2.1388)	72.9661333
4	(-3.6041	-2.7395	-1.8749)	51.7779521
5	(-3.3133	-2.4664	-1.5195)	36.9716654
⋮		⋮		⋮
52	(-3.0254	-2.0324	-1.0565)	0.12774591
53	(-3.0240	-2.0316	-1.0539)	0.11903562
54	(-3.0228	-2.0308	-1.0513)	0.11091931
55	(-3.0218	-2.0301	-1.0488)	0.10335642
56	(-3.0199	-2.0294	-1.0464)	0.09630926

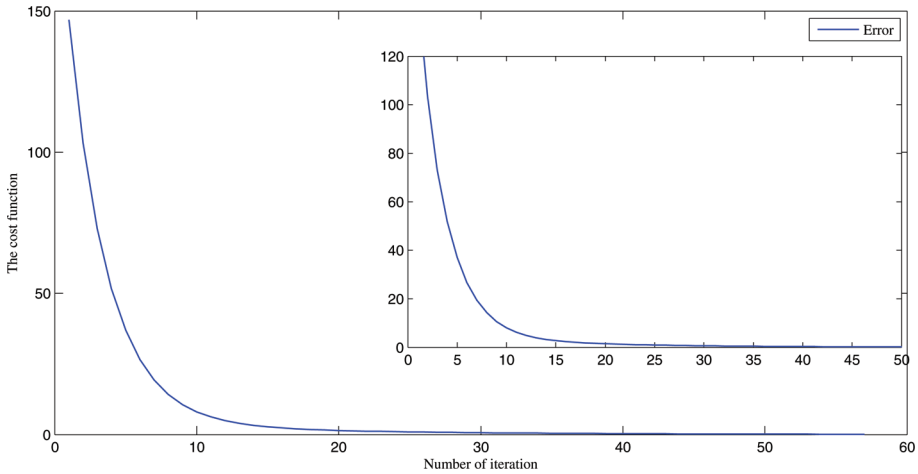

Figure 3: The cost function for Example 2 on the number of iterations.

and $\gamma = 0.001$. Table 2 shows the approximated solution over number of iterations and Fig. 3 shows the accuracy of the solution $x_0(t)$ where t is the number of iterations, its noticeable that by increasing the iteration the cost function goes to zero.

4. Conclusions

In this paper, an architecture of feed-back neural network has been proposed to approximate solution of a FFP with degree one. We suggested a FNN3 model equivalent to the FFP that, the input-output relations were defined by the extension principle. 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

[1] S. Abbasbandy, M. Otadi, Numerical solution of fuzzy polynomials by fuzzy neural network, Appl. Math. Comput. 181 (2006) 1084–1089.

[2] G. Alefeld, J. Herzberger, Introduction to Interval Computations, Academic Press, New York, (1983).

[3] B. Asady, S. Abbasbandy, M. Alavi, Fuzzy general linear systems, Appl. Math. Comput. 169 (2005) 34–40.

[4] J. J. Buckley, Y. Qu, Solving linear and quadratic fuzzy equations, Fuzzy Sets Syst. 35 (1990) 43–59.

[5] R. Fuller, Neural fuzzy systems, Department of Information Technologies, Abo Akademi University. (1995).

[6] Y. Hayashi, J. J. Buckley, E. Czogala, Fuzzy neural network with fuzzy signals and weights, Int. J. Intell. Syst. 8 (1993) 527–537.

[7] H. Ishibuchi, K. Kwon, H. Tanaka, A learning of fuzzy neural networks with triangular fuzzy weghts, Fuzzy Sets Syst. 71 (1995) 277–293.

[8] H. Ishibuchi, H. Okada, H. Tanaka, Fuzzy neural networks with fuzzy weights and fuzzy biases, in: Proc. ICNN 93 (San Francisco), 4 (1993) 1650–1655.

[9] A. Jafarian, S. Measoomynia, Solving fuzzy polynomials using neural nets with a new learning algorithm, Appl. Math. Sci. 5 (2011) 2295–2301.

[10] S. K. Oh, W. Pedrycz, S. B. Roh, Genetically optimized fuzzy polynomial neural networks with fuzzy set-based polynomial neurons, Inform. Sci. 176 (2006) 3490–3519.

[11] C. C. Wang, C. F. Tsai, Fuzzy processing using polynomial bidirectional heteroassociative network, Inform. Sci. 125 (2000) 167–179.

[12] L. A. Zadeh, The concept of a liguistic variable and its application to approximate reasoning: Parts 1-3, Inform. Sci. 8 (1975) 199–249.