Approximated Solutions for Fuzzy Polynomials

Suriana Lasaraiya[1#], **Suzelawati Zenian**[2]

1 Centre of Preparatory Science & Technology, Universiti Malaysia Sabah, Jalan UMS, 88400 Kota Kinabalu, Sabah, MALAYSIA.
2 Faculty of Science and Natural Resources, Universiti Malaysia Sabah, Jalan UMS, 88400 Kota Kinabalu, Sabah, MALAYSIA.
# Corresponding author. E-Mail: suriana@ums.edu.my; Tel: +60168142915.

ABSTRACT Polynomials plays a major role in various fields such as mathematics, statistics, engineering and social science. By using the properties of triangular membership functions, we apply the learning algorithm of fuzzy neural network to solve the fuzzy polynomial equation. The learning algorithm is including the fuzzy coefficient A_i where $i = 1, 2, 3$ and fuzzy output of A_0. In this paper, we are interested in finding solutions for polynomial like $A_1x + A_2x^2 + A_3x^3 = A_0$ for $x \in R$ where A_0, A_1, A_2 and A_3 are fuzzy numbers.

KEYWORDS: fuzzy, polynomials, triangular, fuzzy number, membership function
Received 1 December 2020 **Revised** 20 December 2020 **Accepted** 15 June 2021 **Online** 2 December 2021
© Transactions on Science and Technology
Original Article

INTRODUCTION

Some problems that arise in physics, biology, engineering and many fields in mathematics containing fuzzy polynomials. Some fuzzy polynomials are usually complicated to be solved analytically, then several numerical approaches have been introduced to find the solution of polynomials that is fuzzy polynomials. One of the indirect approaches is Fuzzy Neural Network (FNNs).

Ishibuchi *et al.* (1995), has introduced a cost function that will be applied in the approaches whereby every pair of fuzzy output vector and its corresponding fuzzy target vector. Later on, Buckley & Eslami (1997), applying the idea of FNN in solving fuzzy equation. Then, the same researcher extended the same topics with the same idea in solving problems related to fuzzy polynomials. Abbasandy *et al.* (2006), proposed a logarithm of feed-forward FNN for finding the crisp solutions for the selected fuzzy polynomials. Other researchers as well, Nurhakimah *et al.* (2018) and Behera & Chakraverty (2013) also tried to solve the fuzzy polynomials by using the neural network with a new learning algorithm. The algorithm is an extended principle that was proposed by Zadeh (1975). The input signal was the coefficient of fuzzy equations, while the right-hand side of the FNs is the output target. This learning algorithm was done to find the crisp solution for the triangular fuzzy polynomials.

In this paper, by using fuzzy neural network method, we are interested in finding solutions for polynomial like $A_1x + A_2x^2 + A_3x^3 = A_0$ for $x \in R$ (if exist) where A_0, A_1, A_2 and A_3 are fuzzy numbers. There are two layers in the neural network model, where the coefficients of the left-hand side are the input value, while the right-hand side is the output value. Section 2 will define some preliminaries for defining fuzzy numbers while Section 3 will explain on the learning algorithm being used. Section 4 will show one example using the learning algorithm and finally the conclusion section.

BACKGROUND THEORY

Triangular Fuzzy Number

The definition of triangular fuzzy number of $u = (\gamma, \alpha, \beta)$ for $\alpha > 0$, $\beta > 0$ with their membership functions is given as follows (Zadeh *et al.*, 1965).

$$\mu_u(x) = \begin{cases} \frac{x-\gamma}{\alpha} + 1\,, & \gamma - \alpha \le x \le \gamma \\ \frac{\gamma - x}{\beta} + 1\,, & \gamma \le x \le \gamma + \beta \\ 0\,, & otherwise. \end{cases} \quad (1)$$

A crisp number p can be written as $\mathrm{u_L(r) = u_U(r) = p}$, for $0 \le r \le 1$. Hence, by referring to Equation (1), its parametric form is given by $u_L(r) = \gamma + \alpha(r-1)$ and $u_U(r) = \gamma + \beta(1-r)$.

Operations of Fuzzy Numbers

Fuzzy numbers operations are defined by the study of extension principle in Zadeh (1975). Since the input is fuzzified in the calculations, thus their basic operations such as addition, multiplication and nonlinear mapping are required to define our neural network are given as in Equation (2) and Equation (3).

$$\mu_{M+N}(z) = \max\{\mu_M(x) \wedge \mu_N(y) | z = x + y\} \quad (2)$$

$$\mu_{f(net)}(z) = \max\{\mu_{Net}(x) | z = f(x)\} \quad (3)$$

where *M, N, Net* are fuzzy numbers. $\mu_*(.)$ denotes the membership function of each fuzzy number, $\wedge$is the minimum operator, and $f(x) = x$ is the activation function of output unit of our fuzzy neural network. The *h*-level set of a fuzzy number *X* is defined as $\mathrm{[X]^h = \{x | \mu_X(x) \ge h, x \in R\}}$ for $0 < h \le 1$. We denote $[X]^h$ as $\mathrm{[X]^h = [[X]^h_L, [X]^h_U]}$, since the level sets of fuzzy numbers become closed intervals, whereby $[X]^h_L$ and $\mathrm{[X]^h_U}$ are the lower limit and the upper limit of the h-level set $\mathrm{[X]^h}$, respectively.

The operations of fuzzy number (Equation (4) - (7)) are written for *h*-level sets are as follow, as given by Alefeld & Herzberger (1983).

$$[M]^h + [N]^h = \left[[M]^h_L + [N]^h_L\,, [M]^h_U + [N]^h_U\right], \quad (4)$$

$$f\left([Net]^h\right) = f\left(\left[[Net]^h_L\,, [Net]^h_U\,\right]\right) = \left[f\left([Net]^h_L\right), f\left([Net]^h_U\right)\right], \quad (5)$$

$$w \cdot [M]^h = \left[w \cdot [M]^h_L\,, w \cdot [M]^h_U\right], \qquad if\ w \ge 0, \quad (6)$$

$$w \cdot [M]^h = \left[w \cdot [M]^h_U\,, w \cdot [M]^h_L\right], \qquad if\ w < 0, \quad (7)$$

Fuzzy Polynomials

In this case, we are interested in finding solution for $A_1x + A_2x^2 + A_3x^3 = A_0$ for $x \in R$, when $A_i \in E$, for *i=1,2,3*. The fuzzy neural network in solving our fuzzy polynomials is shown by Ishibuchi *et al.* (1995) as shown in Figure 1.

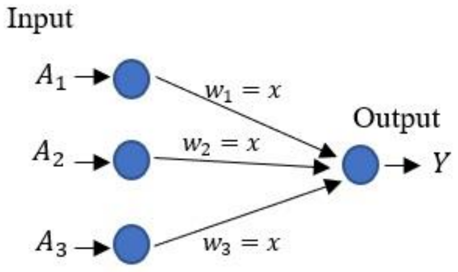

Figure 1. The concept of fuzzy neural network to solve fuzzy polynomials.

Based on Figure 1, the value in the input neuron will be $A_1x + A_2x^2 + A_3x^3 = A_0$, while the output in the output neuron will be $Y = A_1x + A_2x^2 + A_3x^3$.

METHODOLOGY

Input-Output Relation of Each Unit

We will fuzzify two layers feedforward neural network with 3 inputs units with one output unit. Our input vector, target output and the connection weight are crisp numbers. The learning rule will be derived as followed in previous studies, while fuzzy input and fuzzy target will be restricted only within triangular fuzzy numbers. Their relations are defined as in Hayashi *et al.* (1993) and Ishibuchi *et al.* (1995). According to Figure 1. above, a fuzzy input vector of $A = (A_1, A_2, A_3)$ is presented in fuzzy neural network, their relations can be written as in Equations (8) - 10).

Input units:

$$\mathrm{O_i = M_i}\,, \qquad \mathrm{i = 1,2,3.} \tag{8}$$

Output units:

$$\mathrm{Y = f(net),} \tag{9}$$

$$\mathrm{Net} = \textstyle\sum_{j=1}^{3} \mathrm{w_j \cdot O_j}, \tag{10}$$

where M_i and w_j are fuzzy input and crisp weight respectively.

Calculation of Fuzzy Output

The fuzzy output from Equation (8) -(10) are numerically calculated for crisp weights and level sets of fuzzy inputs. The input-output relations for *h*-level sets is as shown as in Equation (11) - (13).

Input units:

$$[\mathrm{O_i}]^{\mathrm{h}} = [\mathrm{M_i}]^{\mathrm{h}}\,, \quad \mathrm{i = 1,2,3.} \tag{11}$$

Output units:

$$[\mathrm{Y}]^{\mathrm{h}} = \mathrm{f}\big([\mathrm{Net}]^{\mathrm{h}}\big), \tag{12}$$

$$[\mathrm{Net}]^{\mathrm{h}} = \textstyle\sum_{j=1}^{n} \mathrm{w_j} \cdot [\mathrm{O_i}]^{\mathrm{h}}, \tag{13}$$

Equation (11) - (13) will give us the *h*-level sets of the fuzzy output *Y*. Based on Equation (4) - (7), the above relations can be written as:

Input units:

$$[\mathrm{O_i}]^{\mathrm{h}} = \big[[\mathrm{O_i}]_{\mathrm{L}}^{\mathrm{h}}\,, [\mathrm{O_i}]_{\mathrm{U}}^{\mathrm{h}}\big] = \big[[\mathrm{A_i}]_{\mathrm{L}}^{\mathrm{h}}\,, [\mathrm{A_i}]_{\mathrm{U}}^{\mathrm{h}}\,\big], \qquad \text{where } \mathrm{i = 1,2,3.} \tag{14}$$

Output unit:

$$[\mathrm{Y}]^{\mathrm{h}} = \big[[\mathrm{Y}]_{\mathrm{L}}^{\mathrm{h}}\,, [\mathrm{Y}]_{\mathrm{U}}^{\mathrm{h}}\big] = \big[\mathrm{f}\big([\mathrm{Net}]_{\mathrm{L}}^{\mathrm{h}}\big), \mathrm{f}\big([\mathrm{Net}]_{\mathrm{U}}^{\mathrm{h}}\big)\big], \tag{15}$$

$$[\mathrm{Net}]^{\mathrm{h}} = \big[[\mathrm{Net}]_{\mathrm{L}}^{\mathrm{h}}, [\mathrm{Net}]_{\mathrm{U}}^{\mathrm{h}}\big] = \Big[\textstyle\sum_{j\in m} \mathrm{w_j} \cdot \big[\mathrm{O_j}\big]_{\mathrm{L}}^{\mathrm{h}} + \sum_{j\in m} \mathrm{w_j} \cdot \big[\mathrm{O_j}\big]_{\mathrm{L}}^{\mathrm{h}}, \sum_{j\in m} \mathrm{w_j} \cdot \big[\mathrm{O_j}\big]_{\mathrm{U}}^{\mathrm{h}} + \sum_{j\in m} \mathrm{w_j} \cdot \big[\mathrm{O_j}\big]_{\mathrm{L}}^{\mathrm{h}}\Big], \tag{16}$$

where $m = \{j | w_j \geq 0\}, c = \{j | w_j < 0\}$ and $m \cup c = \{1, \dots, n\}$.

Learning of Fuzzy Neural Network

We denoted the target output of $\mathrm{A_0}$ by as

$$[\mathrm{A_0}]^{\mathrm{h}} = \big[[\mathrm{A_0}]_{\mathrm{L}}^{\mathrm{h}}\,, [\mathrm{A_0}]_{\mathrm{L}}^{\mathrm{h}}\big],\ h \in [0,1] \tag{17}$$

where the left hand side and right hand side are denoted by $[\mathrm{A_0}]_{\mathrm{L}}^{\mathrm{h}}$ and $[\mathrm{A_0}]_{\mathrm{U}}^{\mathrm{h}}$ respectively of the desired output. As in Jafarian & Jafari (2013) it is stated that the error function for the training pattern and a cost function for each h-level set is defined by:

$$e = \sum_h he(h) \tag{18}$$

$$e(h) = e^L(h) + e^U(h) \tag{19}$$

$$\mathrm{e}^{\mathrm{L}}(\mathrm{h}) = \frac{1}{2}\left(\mathrm{A}_0^{\mathrm{L}}(\mathrm{h})\text{-}\mathrm{Y}^{\mathrm{L}}(\mathrm{h})\right)^2 \tag{20}$$

$$\mathrm{e}^{\mathrm{U}}(\mathrm{h}) = \frac{1}{2}\left(\mathrm{A}_0^{\mathrm{U}}(\mathrm{h})\text{-}\mathrm{Y}^{\mathrm{U}}(\mathrm{h})\right)^2 \tag{21}$$

The weight in Equation (22) is updated by the rules in Equation (10) - (13).

$$\Delta \mathrm{w}_1(\mathrm{t}) = \text{-}\varphi \sum_{\mathrm{h}} \mathrm{h}\frac{\partial \mathrm{e(h)}}{\partial \mathrm{w}_1} + \gamma \cdot \Delta \mathrm{w}_1(\mathrm{t}\text{-}1) \tag{22}$$

where φ is a learning constant, γ is a momentum constant and t indexes number of adjustments. The derivative of Equation (22) is written as shown by Equation (23) - (26):

$$\frac{\partial e(h)}{\partial w_1} = \frac{\partial e^L(h)}{\partial w_1} + \frac{\partial e^U(h)}{\partial w_1} \tag{23}$$

$$\frac{\partial e^L(h)}{\partial w_1} = \frac{\partial e^L(h)}{\partial Y^L} \times \frac{\partial Y^L(h)}{\partial w_1} \tag{24}$$

$$\frac{\partial e^U(h)}{\partial w_1} = \frac{\partial e^U(h)}{\partial Y^U} \times \frac{\partial Y^U(h)}{\partial w_1} \tag{25}$$

$$\frac{\partial e^L(h)}{\partial Y^L} = -\left(A_0^L(h) - Y^L(h)\right), \quad \frac{\partial e^U(h)}{\partial Y^U} = -\left(A_0^U(h) - Y^U(h)\right). \tag{26}$$

If $w \geq 0$, then we have Equation (27) written as

$$\frac{\partial Y^L(h)}{\partial w_1} = A_1^L(h), \quad \frac{\partial Y^U(h)}{\partial w_1} = A_1^U(h), \tag{27}$$

Otherwise, it is written as in Equation (28).

$$\frac{\partial Y^L(h)}{\partial w_1} = A_1^U(h), \quad \frac{\partial Y^U(h)}{\partial w_1} = A_1^L(h). \tag{28}$$

Therefore, if $w_1 \geq 0$, from Equation (22), we have

$$\Delta w_1(t) = \varphi \sum_h h\left[\left(A_0^L(h) - Y^L(h)\right)A_1^L(h) + \left(A_0^U(h) - Y^U(h)\right)A_1^U(h)\right] + \alpha \cdot \Delta w_1(t-1), \tag{29}$$

otherwise, we get

$$\Delta w_1(t) = \varphi \sum_h h\left[\left(A_0^L(h) - Y^L(h)\right)A_1^U(h) + \left(A_0^U(h) - Y^U(h)\right)A_1^L(h)\right] + \alpha \cdot \Delta w_1(t-1). \tag{30}$$

NUMERICAL EXAMPLES

1. Considering the following fuzzy polynomial, as in Abbasbandy (2006):

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

with the exact solution of $\mathrm{x} = \text{-}1$. In this example, we use three input unit and a single output. At $\mathrm{t} = 0$, let $\mathrm{x}_0 = \text{-}1.5$, at $\mathrm{h} = 0.3$, $\varphi = 0.1$, $\gamma = 0.2$. and the calculations is shown as follows:

STEP 1. Find the parametric form using Equation (1).

$[A_0]_L^\alpha = -3.8$	$[A_1]_L^\alpha = -0.7$	$[A_2]_L^\alpha = -1.4$	$[A_3]_L^\alpha = 0.3$
$[A_0]_U^\alpha = 1.8$	$[A_1]_U^\alpha = 0.7$	$[A_2]_U^\alpha = 1.4$	$[A_3]_U^\alpha = 0.7$

STEP 2. Using the value in STEP 1, and apply it in the fuzzy polynomial to find the [NetL] and [NetU].

$$\begin{aligned}[NetL] &= x_0[A_1]_L^\alpha + x_0^2[A_1]_L^\alpha + x_0^3[A_3]_L^\alpha \\ &= (-1.5)(-0.7) + (-1.5)^2(-1.4) + (-1.5)^3(0.3) \\ &= -3.1125\end{aligned}$$

$$\begin{aligned}[NetU] &= x_0[A_1]_U^\alpha + x_0^2[A_1]_U^\alpha + x_0^3[A_3]_U^\alpha \\ &= (-1.5)(0.7) + (-1.5)^2(1.4) + (-1.5)^3(0.7) \\ &= -0.2625\end{aligned}$$

STEP 3

At $t = 0$;

$$\begin{aligned}\Delta x_0(0) = {} & (0.1)(0.3)\big[\big(-3.8 - (-3.1125)\big)(-0.7) + \big(1.8 - (-0.2625)\big)(0.7) \\ & + \big(-3.8 - (-3.1125)\big)(-1.4) + \big(1.8 - (-0.2625)\big)(1.4) + \big(-3.8 - (-3.1125)\big)(0.3) \\ & + \big(1.8 - (-0.2625)\big)(0.7)\big] + 0.2(0) \\ = {} & 0.03[(-0.6875)(1.8)] + [(2.0625)(2.8)] + 0 \\ = {} & 0.1361\end{aligned}$$

Thus, at $t = 0$,

$$\begin{aligned}x_0(1) &= x_0(0) + \Delta x_0(0) \\ &= -1.5 + 0.1361 \\ &= -1.3639\end{aligned}$$

Repeat STEP 2 to STEP 3 for t = 1.

At $t = 1$;

STEP 2

$$[NetL] = -2.4107$$
$$[NetU] = -0.1264$$

STEP 3

$$\Delta x_0(1) = 0.2640$$

Thus, at $t = 1$,

$$\begin{aligned}x_0(2) &= x_0(1) + \Delta x_0(1) \\ &= -1.3639 + 0.2640 \\ &= -1.0999\end{aligned}$$

The process on STEP 2 – STEP 3 will be repeated until we get the approximated solutions. In this example, we stop at $t = 1$ after 2 times iteration only.

2. Considering the following fuzzy polynomial:

$$(0.1,0.25,0.3)x + (-0.2,0.1,1)x^2 + (0.15,0.25,0.35)x^3 = (0.05,0.5,1.65)$$

The process is applied as in Example 1. The results obtained as in Table 1.

Table 1. The results of the iteration process.

t	$x_0(t+1)$	t	$x_0(t+1)$
	0.5	5	0.7469
0	0.5338	6	0.7890
1	0.5769	7	0.8322
2	0.6148	8	0.8547
3	0.6505	9	0.8713
4	0.6984	10	0.8854

CONCLUSION

This learning algorithm of fuzzy neural network for solving the fuzzy polynomial of $A_1x + A_2x^2 + A_3x^3 = A_0$, is proposed. The study is restricted for solving triangular fuzzy polynomial equation. We applied the learning algorithm and showing the number of iterations on solving the polynomial, whereby the number of iterations is affected by the initial value, x_0 , h , φ and γ. We would like to explore more on solving trapezoidal fuzzy polynomial equation in future.

ACKNOWLEDGEMENTS

The work presented in this paper was funded by Universiti Malaysia Sabah under research grant number SLB0193-2019, and is greatly acknowledged.

REFERENCES

[1] Abbasbandy, S. & Otadi, M. 2006. Numerical Solution of Fuzzy Polynomials by Fuzzy Neural Network. *Applied Mathematics and Computation*, 181, 1084-1089.

[2] Buckley, J.J. & Eslami, E. 1997. Neural net solutions to fuzzy problems: The quadratic equation. *Fuzzy Set Systems*, 86, 289-298.

[3] Behera, D. & Chakraverty, S. 2013. Solution to Fuzzy System of Linear Equations with Crisp Coeeficients. *Fuzzy nformation and Engineering*, 2, 205-219.

[4] Hayashi, Y., Buckley, J.J. & Czogala, E. 1993. Fuzzy neural network with fuzzy signals and weights. *International Journal of Intelligent System*, 8**,** 527-537

[5] Ishibuchi, H., Kwon, K. & Tanaka, H. 1995. A learning algorithmn of fuzzy neural networks with triangular fuzzy weights. *Fuzzy Sets and Systems*, 71, 277-293.

[6] Jafarian, A. & Jafari, R. 2013. New Iterative Approach for Solving Fully Fuzzy Polynomials. *International Journal of Fuzzy Mathematics and Systems*, 3(1), 75-83.

[7] Nurhakimah, A. R., Lazim, A. & Ahmad Termimi, A. G. 2018. Learning Algorithm of Fuzzy Neural Network for Solving Trapezoidal Fuzzy Polynomial Equation. *Discovering Mathematics*, 40(1), 1-10.

[8] Zadeh, L.A. 1975. The Concept of a Linguistic Variable and its application to approximate reasoning: Part 1-3. *Informatics Science*, 9(1), 43-80.