Utilizing a new feed-back fuzzy neural network for solving a system of fuzzy equations

A. Jafarian 1*, S. Measoomy Nia 1

(1) Department of Mathematics, Urmia Branch, Islamic Azad University, Urmia, Iran

Abstract
In this paper, we intend to offer a new method based on fuzzy neural networks for finding a real solution of fuzzy equations system. Our proposed fuzzified neural network is a five-layer feed-back neural network that corresponding connection weights to output layer are fuzzy numbers. The proposed architecture of artificial neural network, can get a real input vector and calculates it’s corresponding fuzzy output. In order to find the approximate solution of this fuzzy system that supposedly has a real solution, first a cost function is defined for the level sets of fuzzy output and target output. Then a learning algorithm based on the gradient descent method will be introduced that can adjust the crisp input signals. The proposed method is illustrated by several examples with computer simulations.

Keywords: System of fuzzy equations; Fuzzy feed-back neural networks (FFNN); Cost function

1 Introduction
Fuzzy system are very useful for solving many problems in several applied fields like economics, finance, engineering and physics, because these problems often boil down to the solution of a fuzzy system. In the last years, various approaches for solving fuzzy systems have been reported. Friedman et al. [12] used the embedding method and suggested a general model for solving a fuzzy linear system. Allahviranloo [6, 7, 8] used the iterative Jacobi and Gauss Siedel methods, the adomian method and the successive over-relaxation method, respectively. Theoretical aspects of a fuzzy linear system were discussed by Dubois and Prade [11]. M. Dehgan et al. [10] employed iterative techniques for solving
fully fuzzy linear system. Linear and nonlinear fuzzy systems were solved by [3, 1, 9]. Moreover, some different valid methods for solving fuzzy systems have been developed in the last years. For example, artificial neural network approach has been applied for finding the solution of this kind of problems. Ishibuchi et al. [15] defined a cost function for every pair of fuzzy output vector and its corresponding fuzzy target vector and then proposed a learning algorithm of fuzzy neural networks with triangular and trapezoidal fuzzy weights. Hayashi et al. [14] used the fuzzy delta learning rule. The input-output relation of each unit was defined by the Zadeh extension principle [18]. S. Abbasbandy et al. [4] proffered a structure of feed-forward fuzzy neural networks to find a real solution of a fuzzy polynomials system by introducing a learning algorithm. The main aim of this paper is to construct a new algorithm with the use of feed-back neural network to obtain a approximate real solution for system of fuzzy equations (if exist). It is mentioned that in this fuzzy feed-back neural network, the connection weights and the fuzzy output are fuzzy numbers and also input signals are crisp numbers. So, we say to this neural network FFNN4. The given algorithm enables the FFNN4 to approximate crisp solution of the fuzzy system that has been described in above for any desired degree of accuracy. This paper is organized as follows. First in Section 2, the basic notations and definitions of fuzzy numbers and fuzzy derivative are briefly presented. Section 3 describes how to find a real solution of given fuzzy system using FFNN4. Finally, some examples are collected in Section 4.

2 Preliminaries

This section briefly deals with the foundation of fuzzy numbers. We first presented the notation of fuzzy sets and then we introduced some semantical aspects of fuzzy numbers; In addition, the input-output relation of the employed fuzzy neural network presented in this paper. We started by defining the fuzzy number.

Definition 2.1. A fuzzy number is a fuzzy set $u : \mathbb{R}^1 \rightarrow I = [0, 1]$ which satisfies

(i) $u$ is upper semicontinuous,

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

(iii) There is a real numbers $b, c: a \leq b \leq c \leq d$ for which:

(a) $u(x)$ is monotonically increasing on $[a, b]$,

(b) $u(x)$ is monotonically decreasing on $[c, d]$,

(c) $u(x) = 1, b \leq x \leq c$.

The set of all fuzzy numbers (as given by definition 1) is denoted by $E^1$ [13, 16].

Definition 2.2. A fuzzy number $v$ is a pair $(\underline{v}, \overline{v})$ functions $\underline{v}(r), \overline{v}(r) : 0 \leq r \leq 1$. which satisfy the following requirements:

(i) $\underline{v}(r)$ is a bounded monotonically increasing left continuous function,

(ii) $\overline{v}(r)$ is a bounded monotonically decreasing left continuous function,

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

2
A popular fuzzy number is the triangular fuzzy number \( v = (a, b, c) \) with membership function,

\[
\mu_v(x) = \begin{cases} \frac{x-a}{b-a}, & a \leq x \leq b \\ \frac{c-x}{c-b}, & b \leq x \leq c \\ 0, & \text{otherwise}, \end{cases}
\]

where \( a \leq b \leq c \). Its parametric form is:

\[
[v]^a = a + (b - a)\alpha \quad \text{and} \quad [v]^c = c - (c - b)\alpha, \quad 0 \leq \alpha \leq 1.
\]

### 2.1 Operation on fuzzy numbers

We briefly mention fuzzy number operations defined by the Zadeh extension principle [17, 18].

\[
\mu_{A+B}(z) = \max\{\mu_A(x) \land \mu_B(y) \mid z = x + y\},
\]

\[
\mu_{f(Net)}(z) = \max\{\mu_A(x) \land \mu_B(y) \mid z = xy\},
\]

where \( A \) and \( B \) are fuzzy numbers. \( \mu(\cdot) \) denotes the membership function of each fuzzy number, \( \land \) is the minimum operator, and \( f(x) = x \) is the activation function of output unit of our fuzzy neural network.

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

\[
[A]^\alpha = \{x \mid \mu_A(x) \geq \alpha, \ x \in \mathbb{R}\},
\]

\[
[A]^\alpha = [[A]^\alpha, [A]^\alpha],
\]

where \([A]^\alpha\) and \([A]^\alpha\) are the lower and the upper limits of the \( \alpha \)-level set \([A]^\alpha\), respectively.

From interval arithmetic [5], the above operations on fuzzy numbers are written for the \( \alpha \)-level sets as follows:

\[
[A]^\alpha + [B]^\alpha = [[A]^\alpha, [A]^\alpha] + [[B]^\alpha, [B]^\alpha] = [[A]^\alpha + [B]^\alpha, [A]^\alpha + [B]^\alpha], \tag{2.1}
\]

\[
f([\text{Net}]^\alpha) = f([\text{Net}],[\text{Net}]) = f([\text{Net}]^\alpha), f([\text{Net}]^\alpha), \tag{2.2}
\]

\[
k[A]^\alpha = k[[A]^\alpha, [A]^\alpha] = [k[A]^\alpha, k[A]^\alpha], \quad \text{if } k \geq 0,
\]

\[
k[A]^\alpha = k[[A]^\alpha, [A]^\alpha] = [k[A]^\alpha, k[A]^\alpha], \quad \text{if } k < 0.
\]

For arbitrary \( u = (\underline{u}, \overline{u}) \) and \( v = (\underline{v}, \overline{v}) \) we define addition \((u + v)\) and multiplication by \( k \) as [13, 16]:

\[
\overline{(u + v)}(r) = \overline{u}(r) + \overline{v}(r),
\]

\[
\underline{(u + v)}(r) = \underline{u}(r) + \underline{v}(r),
\]

\[
\overline{(k u)}(r) = k \overline{u}(r), \quad (k u)(r) = k \underline{u}(r), \quad \text{if } k \geq 0,
\]

\[
\overline{(k u)}(r) = k \overline{u}(r), \quad (k u)(r) = k \underline{u}(r), \quad \text{if } k < 0.
\]
2.2 Calculation of fuzzy output in FFNN4

We have given a short review on learning of fuzzified feed-back neural networks. First consider a five-layer FFNN4 with two input units, \( n \) neuron in each hidden layer and one output units. In given structure, we assume that the corresponding connection weights to output layer and target output are triangular fuzzy numbers. In Fig. 1 the input vector \((x_0, y_0)\), the input-output relation of each unit and output \( Y_p \) have been presented. These relations can be written as following:

Input units:
The input neurons make no change in their inputs, so:
\[
o_1 = x_0, \quad o_2 = y_0.
\] (2.3)

First hidden units:
\[
O_{pj} = f_{pj}(o_1), \quad j = 1, \ldots, n.
\]

Second hidden units:
\[
O'_pj = g_{pj}(o_2).
\]

Third hidden units:
\[
U_{pj} = net_{pj}, \quad net_{pj} = O_{pj} \cdot O'_pj.
\]

Output unit:
\[
Y_p = Net_p, \quad Net_p = \sum_{j=1}^{n} U_{pj} \cdot A_{pj}, \quad j = 1, 2, \ldots, n.
\]

Using above equations, we can calculate the \( \alpha \)-level sets of the fuzzy output \( Y_p \). Therefore, from Eqs. (2.1)-(2.2) we can write [5]:
\[
[Y_p]^\alpha = [Net_p]^\alpha,
\] (2.4)
\[
[Net_p]^\alpha = [[Net_p]^\alpha_l, [Net_p]^\alpha_u],
\]
\[
[Net_p]^\alpha_l = \sum_{j \in M} U_{pj} \cdot [A_{pj}]^\alpha_l + \sum_{j \in C} U_{pj} \cdot [A_{pj}]^\alpha_u,
\]
\[
[Net_p]^\alpha_u = \sum_{j \in M} U_{pj} \cdot [A_{pj}]^\alpha_u + \sum_{j \in C} U_{pj} \cdot [A_{pj}]^\alpha_l,
\]
where \( M = \{ j | U_{pj} \geq 0 \}, \quad C = \{ j | U_{pj} < 0 \} \) and \( M \cup C = \{1, \ldots, n\} \).

![Fig. 1. The proposed FFNN4 to solve the system of fuzzy equations.](image-url)
3 system of fuzzy equations

In this section, first the system of fuzzy equations is defined. Then we will concentrate on solving this fuzzy system. Here, we consider the system of fuzzy equations in following form:

\[
\begin{align*}
A_{11}f_{11}(x)g_{11}(y) + \ldots + A_{1n}f_{1n}(x)g_{1n}(y) &= A_{10} \\
A_{21}f_{21}(x)g_{21}(y) + \ldots + A_{2n}f_{2n}(x)g_{2n}(y) &= A_{20},
\end{align*}
\]

(3.5)

where \(A_{10}, A_{11}, \ldots, A_{1n}, A_{20}, A_{21}, \ldots, A_{2n}\) are fuzzy numbers. Moreover, we assume that \(f_{pj}\) and \(g_{pj}\) (for \(p = 1, 2\) and \(j = 1, \ldots, n\)) are real functions of the crisp variables \(x\) and \(y\) (if exist), respectively. For simplify the fuzzy coefficient \(A_{pj}\) will always be considered real triangular fuzzy number. As observed in previous section to get an approximate solution, an architecture of FFNN4 for fuzzy system (3.5) has been given in Fig. 1. The modeling scheme is designed with the simple and versatile fuzzy neural network architecture. Two initial values

3.1 Cost function

At first we assume that the real quantities \(x_0\) and \(y_0\) are initialized at random values for unknown variables \(x\) and \(y\), respectively. In this subsection we intend to introduce how to deduce a learning algorithm to update the crisp parameters \(x_0\) and \(y_0\). Let \(A_{p0}\) be fuzzy target output corresponding to the fuzzy coefficient vector \((A_{p} = A_{p1}, \ldots, A_{pn})\). After presenting these parameters into the proposed network, we calculated the fuzzy output by using relations which were introduced in subsection (2.1). Next we defined a cost function for \(\alpha\)-level sets of fuzzy output \(Y_p\) and it’s corresponding target output \(A_{p0}\) as follows:

\[e^\alpha_p = e^\alpha_{pl} + e^\alpha_{pu},\] (3.6)

where

\[e^\alpha_{pl} = \alpha.\frac{([A_{p0}]^\alpha_l - [Y_p]^\alpha_l)^2}{2},\] \hspace{1cm} (3.7)

\[e^\alpha_{pu} = \alpha.\frac{([A_{p0}]^\alpha_u - [Y_p]^\alpha_u)^2}{2}.\] \hspace{1cm} (3.8)

In the cost function (3.6), \(e^\alpha_{pl}\) and \(e^\alpha_{pu}\) can be viewed as the squared errors for the lower limits and the upper limits of the \(\alpha\)-level sets of the fuzzy output \(Y_p\) and target output \(A_{p0}\), respectively. Then the total error of the given neural network is obtained as [15]:

\[e = \sum_{p=1}^{2} \sum_{\alpha} e^\alpha_p.\] (3.9)

3.1.1 Learning algorithm of the FFNN4

Our main aim is adjusting the parameters \(x_0\) and \(y_0\) with the using of learning algorithm which be introduced in below. For crisp parameter \(y_0\) adjust rule can be written as follows:

\[y_0(t + 1) = y_0(t) + \Delta y_0(t),\] \hspace{1cm} (3.10)

\[\Delta y_0(t) = -\eta \frac{\partial e^\alpha_p}{\partial y_0} + \gamma \Delta y_0(t - 1),\] \hspace{1cm} (3.11)
Step 4: Repeat the following procedures for $p = 1, 2$:

Let

Step 2: initialized at random values.

Step 1: Learning algorithm can be summarized as follows:

Now we have:

If $U_{pj} \geq 0$

otherwise

Now we have:

where $M = \{ j \mid U_{pj} \geq 0 \}$ and $C = \{ j \mid U_{pj} < 0 \}$. Similarly, we can update the parameter $x_0$ with using of relations which applied for updating the input signal $y_0$. Now the learning algorithm can be summarized as follows:

Learning algorithm

Step 1: $\eta > 0$, $\gamma > 0$ and $E_{max} > 0$ 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: Let $t := t + 1$. Repeat Step 5 for $\alpha = \alpha_1, ..., \alpha_m$.

Step 4: Repeat the following procedures for $p = 1, 2$:
[i] Forward calculation: Calculate the $\alpha$-level set of the fuzzy output $Y_p$ by presenting the $\alpha$-level set of the fuzzy coefficients vector $A_p$.

[ii] Back-propagation: Adjust crisp parameters $x_0$ and $y_0$ using the cost function (3.6) for the $\alpha$-level sets of the fuzzy output $Y_p$ and the target output $A_p$.

**Step 5**: Cumulative cycle error is computed by adding the present error to $E$.

**Step 6**: The training cycle is completed. For $E < E_{max}$ terminate the training session. If $E > E_{max}$ then $E$ is set to 0 and we initiate a new training cycle by going back to Step 3.

### 4 Numerical examples

In this section we apply this method for two examples. For each example, the computed values of the approximate solution are calculated over a number of iterations and then the cost function is plotted. In the following simulations, we use the specifications as follows:

1. Values of $\alpha = 0, 0.1, ..., 1$.
2. Learning constant $\eta = 0.001$.
3. Momentum constant $\gamma = 0.001$.
4. Stoping conditions: $E_{max} < 0.001$.

#### Example 4.1

Consider the following system of fuzzy equations:

\[
\begin{align*}
(1, 0, 1) e^{x-1}(y + 1)^2 + (3, 0, 3)\sin(x - 1)\cos(y) &= (1, 0, 1) \\
(1, 2, 4) e^{x-1}(y + 1)^2 + (2, 5, 6)\sin(x - 1)\cos(y) &= (1, 2, 4),
\end{align*}
\]

where $x, y \in \mathbb{R}$ and the exact solution is $x = 1$ and $y = 0$. In this example, we apply the proposed method to approximate the solution of this fuzzy system. The training starts with $x_0 = 0.5$ and $y_0 = 0.5$. Table 1 shows the approximated solutions over number of iterations and Figs. 2 and 3 show the accuracy of the solutions $x_0(t)$ and $y_0(t)$ where $t$ is the number of iterations.

**Table 1**

<table>
<thead>
<tr>
<th>$t$</th>
<th>$x_0(t)$</th>
<th>$y_0(t)$</th>
<th>$e$</th>
<th>$t$</th>
<th>$x_0(t)$</th>
<th>$y_0(t)$</th>
<th>$e$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.60601</td>
<td>0.48232</td>
<td>22.4739</td>
<td>42</td>
<td>0.99626</td>
<td>0.0057441</td>
<td>0.0032690</td>
</tr>
<tr>
<td>2</td>
<td>0.64331</td>
<td>0.44822</td>
<td>18.6951</td>
<td>43</td>
<td>0.99669</td>
<td>0.0050850</td>
<td>0.0025620</td>
</tr>
<tr>
<td>3</td>
<td>0.67040</td>
<td>0.41402</td>
<td>17.1529</td>
<td>44</td>
<td>0.99707</td>
<td>0.0045014</td>
<td>0.0020079</td>
</tr>
<tr>
<td>4</td>
<td>0.69603</td>
<td>0.38032</td>
<td>15.3804</td>
<td>45</td>
<td>0.99740</td>
<td>0.0039848</td>
<td>0.0015735</td>
</tr>
<tr>
<td>5</td>
<td>0.72109</td>
<td>0.34757</td>
<td>13.5776</td>
<td>46</td>
<td>0.99770</td>
<td>0.0035274</td>
<td>0.0012331</td>
</tr>
<tr>
<td>6</td>
<td>0.74515</td>
<td>0.31616</td>
<td>11.8196</td>
<td>47</td>
<td>0.99797</td>
<td>0.0031226</td>
<td>0.0009663</td>
</tr>
</tbody>
</table>
Example 4.2. Let fuzzy system
\[
\begin{align*}
(1, 2, 3)xy + (2, 3, 4)e^{x+1} \cos(y - 1) &= (-1, 1, 3) \\
(2, 3, 4)xy^2 + (3, 4, 5)x^3 \sin(y - 1) &= (-4, -3, -2),
\end{align*}
\]
with the exact solutions \( x = -1 \) and \( y = 1 \). We trained the fuzzy neural network as described in last example. Before starting calculations, we assume that \( x_0 = -1.5 \) and \( y_0 = 0.5 \). Numerical result can be found in table 2. Figs. 4 and 5 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 4.2 case 1.
Example 4.3. We consider the following fuzzy system

\[
\begin{align*}
2 \sin \left( \frac{\pi}{2} (x + 1) \right) \cos^3(y) + (1, 3, 7) \tan \left( \frac{\pi}{4} (x + 1) \right) \ln^{e(y+1)^2} &= (3, 5, 9) \\
(3, 7, 8) \sin \left( \frac{\pi}{2} (x + 1) \right) \cos^3(y) + 3 \tan \left( \frac{\pi}{4} (x + 1) \right) \ln^{e(y+1)^2} &= (6, 10, 11),
\end{align*}
\]

(4.15)

with the exact solutions \( x = 0 \) and \( y = 0 \). We trained the fuzzy neural network as described in previous examples. Before starting calculations, we assume that \( x_0 = 0.5, \ y_0 = -0.5, \)
\( \eta = 0.001 \) and \( \gamma = 0.001 \). Numerical result can be found in Table 3. Figs. 6 and 7 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 4.3 case 1.

<table>
<thead>
<tr>
<th>( t )</th>
<th>( x_0(t) )</th>
<th>( y_0(t) )</th>
<th>( e )</th>
<th>( t )</th>
<th>( x_0(t) )</th>
<th>( y_0(t) )</th>
<th>( e )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.06917</td>
<td>-0.04241</td>
<td>0.1116537</td>
<td>160</td>
<td>0.03242</td>
<td>-0.02320</td>
<td>0.0010458</td>
</tr>
<tr>
<td>2</td>
<td>0.06731</td>
<td>-0.04392</td>
<td>0.022291</td>
<td>161</td>
<td>0.03034</td>
<td>-0.02115</td>
<td>0.0010358</td>
</tr>
<tr>
<td>3</td>
<td>0.06663</td>
<td>-0.04367</td>
<td>0.016967</td>
<td>162</td>
<td>0.02826</td>
<td>-0.02280</td>
<td>0.0010259</td>
</tr>
<tr>
<td>4</td>
<td>0.06598</td>
<td>-0.04322</td>
<td>0.016318</td>
<td>163</td>
<td>0.02600</td>
<td>-0.02225</td>
<td>0.0010162</td>
</tr>
<tr>
<td>5</td>
<td>0.06555</td>
<td>-0.04287</td>
<td>0.015722</td>
<td>164</td>
<td>0.02510</td>
<td>-0.02159</td>
<td>0.0010067</td>
</tr>
<tr>
<td>6</td>
<td>0.06473</td>
<td>-0.04233</td>
<td>0.015159</td>
<td>165</td>
<td>0.02453</td>
<td>-0.02094</td>
<td>0.00099727</td>
</tr>
</tbody>
</table>

**Fig. 6.** The cost function for Example 4.3.

**Fig. 7.** Convergence of the calculated solutions for Example 4.3.
5 Conclusions

In this paper, an architecture of feed-back neural networks has been proposed for approximating solution of a fuzzy equations system. We suggested a FFNN4 model equivalent to the system 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. Finally, we can conclude that for this kind of problems the introduced model presents good performance than feed-forward samples. Because the speed of convergence is increased which depends on number of computations.

Acknowledgment

This paper is based on the Masters project at the Department of Mathematics, Urmia Branch Islamic Azad University of Iran, under graduate research fellowship and fundamental research grant scheme. The respected anonymous referees have carefully reviewed this paper. As a result of their careful analysis, our paper has been improved. The authors would like to express their thankfulness to them for their helpful constructive comments.

References


http://dx.doi.org/10.1016/j.amc.2004.10.042


http://dx.doi.org/10.1016/0165-0114(80)90004-4

http://dx.doi.org/10.1016/S0165-0114(96)00270-9

http://dx.doi.org/10.1016/0165-0114(86)90026-6

http://dx.doi.org/10.1002/int.4550080405

http://dx.doi.org/10.1016/0165-0114(94)00281-B

http://dx.doi.org/10.1016/0022-247X(78)90045-8

http://dx.doi.org/10.1016/j.ins.2005.01.017