Fuzzy Fredholm integro-differential equations with artificial neural networks

Maryam Mosleh1, Mahmood Otadi1

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

Abstract
In this paper, we use parametric form of fuzzy number, then feed-forward neural network is presented for obtaining approximate solution for fuzzy Fredholm integro-differential equation of the second kind. This paper presents a method based on neural networks and Newton-Cotes methods with positive coefficient. The ability of neural networks in function approximation is our main objective. The proposed method is illustrated by solving some numerical examples.

Keywords: Fuzzy integro-differential equations; Artificial neural networks

1 Introduction

The solutions of integral equations have a major role in the field of science and engineering. A physical even can be modelled by the differential equation, an integral equation. Since few of these equations cannot be solved explicitly, it is often necessary to resort to numerical techniques which are appropriate combinations of numerical integration and interpolation [12, 32]. There are several numerical methods for solving linear Volterra integral equation [18, 41] and system of nonlinear Volterra integral equations [14]. Kauthen in [28] used a collocation method to solve the Volterra-Fredholm integral equation numerically. Borzabadi and Fard in [16] obtained a numerical solution of nonlinear Fredholm integral equations of the second kind.

The concept of fuzzy numbers and fuzzy arithmetic operations were first introduced by Zadeh [43], Dubois and Prade [20]. We refer the reader to [26] for more information on fuzzy numbers and fuzzy arithmetic. The topics of fuzzy integral equations (FIE)
which growing interest for some time, in particular in relation to fuzzy control, have been rapidly developed in recent years. The fuzzy mapping function was introduced by Chang and Zadeh [17]. Later, Dubois and Prade [21] presented an elementary fuzzy calculus based on the extension principle also the concept of integration of fuzzy functions was first introduced by Dubois and Prade [21]. Babolian et al. and Abbasbandy et al. in [3, 11] obtained a numerical solution of linear Fredholm fuzzy integral equations of the second kind. Allahviranloo et al. in [6] presented a new method for solving fuzzy integrodifferential equation under generalized differentiability.

Another group of researchers tried to extend some numerical methods to solve fuzzy differential equations (FDEs) [1, 8, 38] such as Runge-Kutta method [2], Adomian method [10], predictor-corrector method and multi-step methods [7].

Fuzzy neural network have been extensively studied [15, 19] and recently, successfully used for solving fuzzy polynomial equation and systems of fuzzy polynomials [4, 5], approximate fuzzy coefficients of fuzzy regression models [33, 34, 35], approximate solution of fuzzy linear systems and fully fuzzy linear systems [36, 39]. Lagaris et al. in [31] used multilayer perceptron to estimate the solution of differential equation. Their neural network model was trained over an interval (over which the differential equation must be solved), so the inputs of the neural network model were the training points. The comparison of their method with the existing numerical method shows that their method was more accurate and the solution had also more generalizations. Recently Effati et al. in [22] and Mosleh and Otadi [37] used artificial neural networks to solve fuzzy ordinary differential equations. But in this paper, we extend the artificial neural networks to solve integro-differential equations. The ability of neural networks in function approximation is our main objective. In this paper, we present a novel and very simple numerical method based upon neural networks for solving fuzzy linear Fredholm integro-differential equations of the second kind

\[ X'(s) = y(s) + \lambda \int_a^b k(s, t)X(t)dt. \]

2 Preliminaries

In this section the basic notations used in fuzzy operations are introduced. We start by defining the fuzzy number.

**Definition 2.1.** [30] A fuzzy number is a fuzzy set \( u : \mathbb{R} \rightarrow I = [0, 1] \) such that

i. \( u \) is upper semi-continuous;

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

iii. There are real numbers \( b \) and \( c \), \( a \leq b \leq c \leq d \), for which

1. \( u(x) \) is monotonically increasing on \([a, b]\),

2. \( u(x) \) is monotonically decreasing on \([c, d]\),

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

The set of all the fuzzy numbers (as given in Definition (2.1) is denoted by \( E^1 \).

An alternative definition which yields the same \( E^1 \) is given by Kaleva [27].

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

i. \( u(r) \) is a bounded monotonically increasing, left continuous function on \((0, 1]\) and right continuous at 0;
ii. \( u(r) \) is a bounded monotonically decreasing, left continuous function on \((0, 1]\) and right continuous at 0;

iii. \( u(r) \leq \overline{u}(r), 0 \leq r \leq 1 \).

A crisp number \( r \) is simply represented by \( u(r) = u(r) = r; 0 \leq r \leq 1 \). The set of all the fuzzy numbers is denoted by \( E^1 \). This fuzzy number space as shown in [42], can be embedded into the Banach space \( B = \overline{C}[0, 1] \times \overline{C}[0, 1] \).

**Definition 2.3.** [27] For arbitrary \( u = (u(r), \overline{u}(r)), v = (v(r), \overline{v}(r)) \), we say that \( u = v \) if and only if \( u = v \) and \( \overline{u} = \overline{v} \).

For arbitrary \( u = (u(r), \overline{u}(r)), v = (v(r), \overline{v}(r)) \) and \( k \in \mathbb{R} \) we define addition and multiplication by \( k \) as

\[
\begin{align*}
(u + v)(r) &= (u(r) + v(r)), \\
(u + v)(r) &= (\overline{u}(r) + \overline{v}(r)), \\
k u(r) &= k u(r), \overline{k u}(r) = k \overline{u}(r), \text{ if } k \geq 0, \\
k u(r) &= k \overline{u}(r), \overline{k u}(r) = k u(r), \text{ if } k < 0.
\end{align*}
\]

**Definition 2.4.** [24] For arbitrary fuzzy numbers \( u, v \), we use the distance

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

and it is shown that \((E^1, D)\) is a complete metric space [40].

**Definition 2.5.** [23, 24] Let \( f : [a, b] \to E^1 \), for each partition \( P = \{t_0, t_1, \ldots, t_n\} \) of \([a, b]\) and for arbitrary \( \xi_i \in [t_{i-1}, t_i], 1 \leq i \leq n \) suppose

\[
R_P = \sum_{i=1}^{n} f(\xi_i)(t_i - t_{i-1}),
\]

\[
\Delta := \max\{|t_i - t_{i-1}|, i = 1, 2, \ldots, n\}.
\]

The definite integral of \( f(t) \) over \([a, b]\) is

\[
\int_{a}^{b} f(t)dt = \lim_{\Delta \to 0} R_P
\]

provided that this limit exists in the metric \( D \).

If the fuzzy function \( f(t) \) is continuous in the metric \( D \), its definite integral exists [24] and also,

\[
\int_{a}^{b} f(t; r)dt = \int_{a}^{b} f(t; r)dt,
\]

\[
\int_{a}^{b} f(t; r)dt = \int_{a}^{b} f(t; r)dt.
\]

**Definition 2.6.** 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 2.7. A function \( f : (a, b) \rightarrow E^1 \) is called \( H \)-differentiable at \( t \in (a, b) \) if, for \( h > 0 \) sufficiently small, there exist the \( H \)-differences \( f(t + h) - f(t) \), \( f(t) - f(t - h) \), and an element \( f'(t) \in E^1 \) such that:

\[
\lim_{h \to 0^+} D\left( \frac{f(t + h) - f(t)}{h} \right) = \lim_{h \to 0^+} D\left( \frac{f(t) - f(t - h)}{h} \right), f'(t) = 0.
\]

Then \( f'(t) \) is called the fuzzy derivative of \( f \) at \( t \).

3 Fuzzy integro-differential equation

The linear Fredholm integro-differential equations [25]

\[
X'(s) = y(s) + \lambda \int_a^b k(s, t)X(t)dt, \quad X(s_0) = X_0,
\]

where \( \lambda > 0 \), \( k \) is an arbitrary given kernel function over the square \( a \leq s, t \leq b \) and \( y(s) \) is a given function of \( s \in [a, b] \). If \( X \) is a fuzzy function, \( y(s) \) is a given fuzzy function of \( s \in [a, b] \) and \( X' \) is the fuzzy derivative (according to Definition (2.7)) of \( X \), this equation may only possess fuzzy solution. Sufficient for the existence equation of the second kind, are given in [13].

Let \( X(s) = (X(s; r), \overline{X}(s; r)) \) is a fuzzy solution of Eq.(3.1), therefore by Definition (2.3), Definition (2.5) and Definition (2.7) we have the equivalent system

\[
X'(s) = y(s) + \lambda \int_a^b k(s, t)X(t)dt, \quad X(s_0) = X_0,
\]

\[
\overline{X}'(s) = \overline{y}(s) + \lambda \int_a^b \overline{k}(s, t)\overline{X}(t)dt, \quad \overline{X}(s_0) = \overline{X}_0
\]

which possesses a unique solution \((\underline{X}, \overline{X}) \in B\) which is a fuzzy function, i.e. for each \( s \), the pair \((X(s; r), \overline{X}(s; r))\) is a fuzzy number, therefore each solution of Eq.(3.1) is a solution of system (3.2) and conversely also Eq.(3.1) and system (3.2) are equivalent.

The parametric form of Eqs.(3.2) is given by

\[
X'(s, r) = \underline{y}(s, r) + \lambda \int_a^b k(s, t)\underline{X}(t, r)dt, \quad \underline{X}(s_0) = \underline{X}_0(r),
\]

\[
\overline{X}'(s, r) = \overline{y}(s, r) + \lambda \int_a^b \overline{k}(s, t)\overline{X}(t, r)dt, \quad \overline{X}(s_0) = \overline{X}_0(r)
\]

for \( r \in [0, 1] \), suppose \( k(s, t) \) be continuous in \( a \leq s \leq b \) and for fix \( t, k(s, t) \) changes its sign in finite points as \( s_i \) where \( x_i \in [a, s_1] \). For example, let \( k(s, t) \) be nonnegative over \([a, s_1]\) and negative over \([s_1, b]\), therefore we have

\[
X'(s, r) = \underline{y}(s, r) + \lambda \int_a^{s_1} k(s, t)\underline{X}(t, r)dt + \lambda \int_{s_1}^b k(s, t)\overline{X}(t, r)dt, \quad \underline{X}(s_0) = \underline{X}_0(r),
\]

\[
\overline{X}'(s, r) = \overline{y}(s, r) + \lambda \int_a^{s_1} k(s, t)\overline{X}(t, r)dt + \lambda \int_{s_1}^b k(s, t)\underline{X}(t, r)dt, \quad \overline{X}(s_0) = \overline{X}_0(r).
\]

In most cases, however, analytical solution to Eq.(3.3) may not be found and a numerical approach must be considered.
4 Function approximation

The use of neural networks provides solutions with very good generalizability (such as differentiability). On the other hand, an important feature of multi-layer perceptrons is their utility to approximate functions, which leads to a wide applicability in most problems.

In this paper, the function approximation capabilities of feed-forward neural networks is used by expressing the trial solutions for a system (3.3) as the sum of two terms. The first term satisfies the initial conditions and contains no adjustable parameters. The second term involves a feed-forward neural network to be trained so as to satisfy the integro-differential equations. Since it is known that a multilayer perceptron with one hidden layer can approximate any function to arbitrary accuracy, the multilayer perceptron is used as the type of the network architecture.

If \( X_T(s, r, \bar{p}) \) is a trial solution for the first equation in system (3.3) and \( \bar{X}_T(s, r, \bar{p}) \) is a trial solution for the second equation in the system (3.3) where \( \bar{p} \) and \( \bar{p} \) are adjustable parameters, (indeed \( X_T(s, r, \bar{p}) \) and \( \bar{X}_T(s, r, \bar{p}) \) are approximations of \( X(s, r) \) and \( \bar{X}(s, r) \) respectively) thus the problem of finding the approximated solutions for (3.3) over some collocation point in \([a, b]\) is equivalent to calculate the functionals \( X_T \) and \( \bar{X}_T \) that satisfies the following constrained optimization problem [29]:

\[
\begin{align*}
\min_{\bar{p}} \sum_{i=1}^{M} \{(X'_T(s_i, r, \bar{p}) - g(s_i, r) - F(s_i, r, \bar{p}))^2 + (\bar{X}_T(s_i, r, \bar{p}) - \bar{g}(s_i, r) - \bar{F}(s_i, r, \bar{p}))^2 \}, \\
X_T(s_0, r, \bar{p}) = X_0(r), \quad \bar{X}_T(s_0, r, \bar{p}) = \bar{X}_0(r)
\end{align*}
\]

(4.4)

where \( \bar{p} = (p, \bar{p}) \) contain all adjustable parameters (weights of input and output layers and biases) and

\[
\begin{align*}
F(s, r, \bar{p}) &= \lambda \int_a^b k(s, t)X_T(t, r, \bar{p}) dt, \\
\bar{F}(s, r, \bar{p}) &= \lambda \int_a^b k(s, t)\bar{X}_T(t, r, \bar{p}) dt.
\end{align*}
\]

In general we cannot be able to carry out analytically the integrations, involved. In this case we naturally turn to numerical quadrature. We introduce a quadrature rule \( R \) for the interval \([a, b]\) with positive weights \( w_j \) and \( N \) nodes \( t_j \), i.e.,

\[
Rf = \sum_{j=1}^{N} w_j f(t_j) = If - Ef = \int_a^b f(t) dt - Ef,
\]

where \( Ef \) is the error. If we first ignore the error of this quadrature rule then the Eq. (4.4) is replaced by the approximate equation

\[
\begin{align*}
\min_{\bar{p}} \sum_{i=1}^{M} \{(X'_T(s_i, r, \bar{p}) - g(s_i, r) - \lambda \sum_{j=1}^{N} w_j k(s_i, t_j)X_T(t_j, r, \bar{p}))^2 + (\bar{X}_T(s_i, r, \bar{p}) - \bar{g}(s_i, r) - \lambda \sum_{j=1}^{N} w_j k(s_i, t_j)\bar{X}_T(t_j, r, \bar{p}))^2 \}, \\
X_T(s_0, r, \bar{p}) = X_0(r), \quad \bar{X}_T(s_0, r, \bar{p}) = \bar{X}_0(r)
\end{align*}
\]

(4.5)

Each trial solution \( X_T \) and \( \bar{X}_T \) employs one feed-forward neural network for which the corresponding networks are denoted by \( N \) and \( \bar{N} \), with adjustable parameters \( p \) and \( \bar{p} \), respectively. The related trial functions will be in the form [22]:

\[
\begin{align*}
X_T(s, r, \bar{p}) &= X(s_0, r) + (s - s_0)N(s, r, \bar{p}), \\
\bar{X}_T(s, r, \bar{p}) &= \bar{X}(s_0, r) + (s - s_0)\bar{N}(s, r, \bar{p}),
\end{align*}
\]

(4.6)
where \( \mathcal{N} \) and \( \overline{\mathcal{N}} \) are single-output feed-forward neural networks with adjustable parameters \( p \) and \( \overline{p} \), respectively. Here \( s \) and \( r \) are the network inputs. This solutions by intention satisfies the initial condition in (4.6). According to (4.6) it is straightforward to show that:

\[
\begin{align*}
X'_T(s, r, p) &= \mathcal{N}(s, r, p) + (s - s_0) \frac{\partial \mathcal{N}}{\partial s}, \\
X'_T(s, r, \overline{p}) &= \mathcal{N}(s, r, \overline{p}) + (s - s_0) \frac{\partial \mathcal{N}}{\partial s}.
\end{align*}
\]  

(4.7)

Now consider a multilayer perceptron having one hidden layer with \( H \) sigmoid units and a linear output unit (Fig. 1, Fig. 2).

Here we have:

\[
\begin{align*}
\mathcal{N} &= \sum_{i=1}^{H} v_i \sigma(z_i), \quad z_i = w_{1i} s + w_{2i} r + b_i, \\
\overline{\mathcal{N}} &= \sum_{i=1}^{H} \overline{v}_i \sigma(\overline{z}_i), \quad \overline{z}_i = \overline{w}_{1i} s + \overline{w}_{2i} r + \overline{b}_i,
\end{align*}
\]  

(4.8)

where \( \sigma(z) \) is the sigmoid transfer function. The following is obtained:

\[
\begin{align*}
\frac{\partial \mathcal{N}}{\partial s} &= \sum_{i=1}^{H} v_i w_{1i} \sigma'(z_i), \\
\frac{\partial \overline{\mathcal{N}}}{\partial s} &= \sum_{i=1}^{H} \overline{v}_i \overline{w}_{1i} \sigma'(\overline{z}_i),
\end{align*}
\]
where $\sigma'(z_i)$ is the first derivative of the sigmoid function. Also there are many choices for the sigmoid function, here we choose $\sigma(z) = 1/(1 + e^{-z})$ since it is possible to derive all the derivatives of $\sigma(z)$ in terms of the sigmoid function itself. i.e.

$$\sigma'(z) = -\sigma^2(z) + \sigma(z).$$

5 Example

To illustrate the technique proposed in this paper, consider the following example. For each fuzzy numbers, we use $r = 0, 0.1, \ldots, 1$, where we calculate the accuracy of the method by Eq. (4.5). In the computer simulation of this section, we use the $H = 10$ sigmoid units in the hidden layer.

Example 5.1. Consider the following fuzzy linear Fredholm integro-differential equation

$$X'(s) = (0.5 + 0.5r, 2 - r)(e^s - s) + \int_0^1 tsX(t)dt,$$
$$X(0) = (0.5 + 0.5r, 2 - r); \quad 0 \leq r \leq 1, \quad 0 \leq s, t \leq 1.$$

The exact solution in this case is given by

$$X = (0.5 + 0.5r, 2 - r)e^s.$$ 

The trial functions for this problem are

$$X_T(s; r) = (0.5 + 0.5r) + s \sum_{i=1}^{H} \frac{v_i}{1 + e^{-w_1^i r - w_2^i s - \beta_i^1}},$$

$$X_T(s; r) = (2 - r) + s \sum_{i=1}^{H} \frac{v_i}{1 + e^{-w_1^i r - w_2^i s - \beta_i^2}}.$$

The exact and obtained solution of fuzzy linear Fredholm integro-differential equation in this example at $s = 1$ are shown in Figure 3, also the error by Eq. (4.5) is $1.124331e^{-4}$.  

![Fig. 3. The exact and approximate solution for example 5.1.](image-url)
Figs. 4-7 show the convergence property of the computed values of the weights.

**Fig. 4.** Convergence of the weights $w_{11}$ for example 5.1.

**Fig. 5.** Convergence of the weights $w_{12}$ for example 5.1.
6 Conclusion

Solving fuzzy integro-differential equation (FIDE) by using universal approximators (UA), that is, neural network model (NNM) is presented in this paper.

In this paper, the original fuzzy integro-differential equation is replaced by two parametric linear Fredholm integro-differential equations which are then solved numerically using UAM. The main reason for using neural networks was their applicability in function approximation. Our computer simulation in this paper were performed for three-layer feedforward neural networks. Since we had good simulation result even from three-layer
neural networks, we do not think that the extension of our NNM to neural networks with more than three layers is an attractive research direction.

References


http://dx.doi.org/10.1016/S0165-0114(98)00355-8

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


[26] A. Kaufmann and M.M. Gupta, Introduction Fuzzy Arithmetic, Van Nostrand Rein-

http://dx.doi.org/10.1016/0165-0114(87)90029-7

[28] J.P. Kauthen, Continuous time collocation method for Volterra-Fredholm integral
http://dx.doi.org/10.1007/BF01396646

[29] D. R. Kincaid, E.W. Cheney, Numerical analysis: Mathematics of scientific comuting,


http://dx.doi.org/10.1137/1.9781611970852

[33] M. Mosleh, M. Otadi and S. Abbasbandy, Evaluation of fuzzy regression models
by fuzzy neural network, Journal of Computational and Applied Mathematics, 234
http://dx.doi.org/10.1016/j.cam.2010.01.046

[34] M. Mosleh, M. Otadi and S. Abbasbandy, Fuzzy polynomial regression with fuzzy
http://dx.doi.org/10.1016/j.apm.2011.04.039

http://dx.doi.org/10.1007/s00521-011-0698-z

[36] M. Otadi and M. Mosleh, Simulation and evaluation of dual fully fuzzy linear systems
http://dx.doi.org/10.1016/j.apm.2011.04.008

[37] M. Mosleh and M. Otadi, Simulation and evaluation of fuzzy differential equations
http://dx.doi.org/10.1016/j.asoc.2012.03.041
http://dx.doi.org/10.1007/s00521-012-0913-6

http://dx.doi.org/10.1007/s00500-010-0685-9

http://dx.doi.org/10.1016/0022-247X(86)90093-4

http://dx.doi.org/10.1016/j.apm.2010.02.034

http://dx.doi.org/10.1016/0165-0114(91)90030-T

http://dx.doi.org/10.1016/0020-0255(75)90036-5