ORIGINAL PAPER

A PROPOSAL FOR FUZZY NEURAL NETWORK IN GENERALIZED FUZZY NUMBERS FRAMEWORK USING FUNCTION PRINCIPLE

DRAGOS AROTARITEI[1], MARIANA ROTARIU[1], MIHAI ILEA[1*], MARIUS TURNEA[1]

Manuscript received: 02.07.2019; Accepted paper: 28.08.2019;
Published online: 30.09.2019.

Abstract. *Fuzzification of multilayer perceptron is proposed using generalized fuzzy numbers and an extension for the activation function of fuzzy neurons. All the operations take place in the generalized fuzzy numbers framework using the function principle. A new learning based on gradient methods is proposed and experimental results are also presented*

Keywords**:** *Fuzzy neural networks, Multilayer perceptron, Function principle, Triangular fuzzy Numbers, Fuzzy neuron.*

1. INTRODUCTION

Fuzzy numbers (FN), triangular fuzzy numbers (TFN), trapezoidal fuzzy numbers (TrFN) and applications are subject to intensive research. Definition of FNs with various shapes (symmetric-triangular, triangular, trapezoidal, Cauchy, exponential) along with basic operations are dealt with in many papers [1, 2]. Basic operations in a fuzzy framework, that is addition, subtraction, multiplication and division, can produce a FN with a modified shape. It is known that multiplication of two TFNs or multiplication of two TrFN have as result a FN with modified shape. In order to better approximate the results, the solution usually proposed is to make operations at each level of α-cut [2].

There are many learning algorithms, some of them based on gradient methods, which approximate iteratively the parameters used by fuzzy neural networks (FNNs). Fuzzification is used for various parameters: weights, inputs, outputs, operations etc. [3, 4]. Very few papers deal with complete fuzzification of fuzzy neural networks: parameters, variables and operations [5-7]. One important question is the convergence of the learning algorithm. The gradient method applied to error minimization for crisp neural networks extended to FNNs generally guarantees a local minima but it is not sure that this point is also a global minima. There are various techniques available to overcome this drawback, especially the back propagation algorithm [8]. Another question is whether fuzzy neural networks can approximate any arbitrary continuous fuzzy functions. The general response is affirmative [9, 10].

All FNNs in various architectures use FNs. To the best of our knowledge, no FNNs with generalized fuzzy numbers (GFNs) have been proposed yet. There are many papers that deal with GFNs [11-13], but very few applications in which GFNs play an important role. A solution to solve transportation problem in a given fuzzy environment is proposed in [13].

[1] "Gr. T. Popa" University of Medicine and Pharmacy, Department of Medical Biosciences, 700115 Iasi, Romania. E-mail: ileamihai2004@yahoo.com; mihail.ilea@umfiasi.ro.

The cost of transportation costs is calculate using generalized trapezoidal fuzzy numbers (GTN) and an algorithm that minimize the cost of transportation with restrictions (constraints) is proposed in [13]. An interesting subject of research is fuzzy linear systems (especially when are used TrFN) [14]. The matrix pf coefficient that describes the system is partitioned into submatrices having TrFN matrices [14].

We suggest the use of FNNs that have fuzzy neurons and all the arithmetic operations in these FNNs are part of the generalized trapezoidal number framework.

2. MATERIALS AND METHODS

2.1. GENERALIYED FUZZY NUMBERS AND FUNCTION PRINCIPLE

A generalized fuzzy number (GFN) is a fuzzy set $\tilde{A}$ on $\Re$ and its membership $\mu_{\tilde{A}(x)}$ satisfies the following conditions:

1. Any α-level set of $\tilde{A}$ denoted by $A_\alpha=\{x|\ \mu_{\tilde{A}(x)} \geq \alpha\}$ is a closed interval;
2. $\exists x \in R$, such as $\mu_{\tilde{A}(x)} = w$.

According to [9, 10], a trapezoidal generalized fuzzy number (TGFN) can be represented by 5-uple $\tilde{A} = (a,b,c,d;h)$, where we have the relations $a \leq b \leq c \leq d$. If h = 1, we are in the traditional fuzzy number framework and the operations with these TFNs are mentioned in [1-5]. In a particular case, when $b = c$, conduct to a triangular generalized fuzzy number (TrGFN) and arithmetic operations [8, 9] proposed for TGFN are simplified.

Let's denote by $\tilde{A}_1 = (a_1,b_1,c_1,d_1;h_1)$ and $\tilde{A}_2 = (a_2,b_2,c_2,d_2;h_2)$ two TGFNs, and $* = \{+,-\}$ the classic arithmetic operations addition and subtraction between two real numbers and $\tilde{*} = \{\oplus,\tilde{-}\}$ the corresponding operations using function principle [9, 10]). The basic arithmetic operations between two TGFNs using the function principle are described in eq. (1)-(5) [9, 10]).

$$\tilde{B} = \tilde{A}_1 \tilde{*} \tilde{A}_2 = (a_1,b_1,c_1,d_1;h_1) * (a_2,b_2,c_2,d_2;h_2) = (a,b,c,d;h) \quad (1)$$

$$a = a_1 * a_2,\ b = b_1 * b_2,\ c = a_1 * a_2,\ a = a_1 * a_2,\ a = a_1 * a_2,\ h = \min(h_1,h_2) \quad (2)$$

The multiplication and division between two GFNs have a different formalism. The multiplication between two GTFNs is given by (Fig. 1):

$$\tilde{B} = \tilde{A}_1 \otimes \tilde{A}_2 = (a_1,b_1,c_1,d_1;h_1) \otimes (a_2,b_2,c_2,d_2;h_2) = (a,b,c,d;\min(h_1,h_2)) \quad (3)$$

where

$$\begin{cases} a = \min(a_1 \times a_2, a_1 \times d_2, d_1 \times a_2, d_1 \times d_2) \\ b = \min(b_1 \times b_2, b_1 \times c_2, c_1 \times b_2, c_1 \times c_2) \\ c = \max(b_1 \times b_2, b_1 \times c_2, c_1 \times b_2, c_1 \times c_2) \\ d = \max(a_1 \times a_2, a_1 \times d_2, d_1 \times a_2, d_1 \times d_2) \end{cases} \quad (4)$$

$$a_1 \le b_1 \le c_1 \le d_1, \quad a_2 \le b_2 \le c_2 \le d_2, \quad a \le b \le c \le d \quad (5)$$

The division between two GFNs is as shown in [9, 10], but this operation is not involved in fuzzification of proposed neural network. Let's denote by $\tilde{A} = (a,b,c,d;h)$ a TGFN. In case all numbers $\{a, b, c, d\}$ are positive, the equations (4)-(5) can be written in a simplified form.

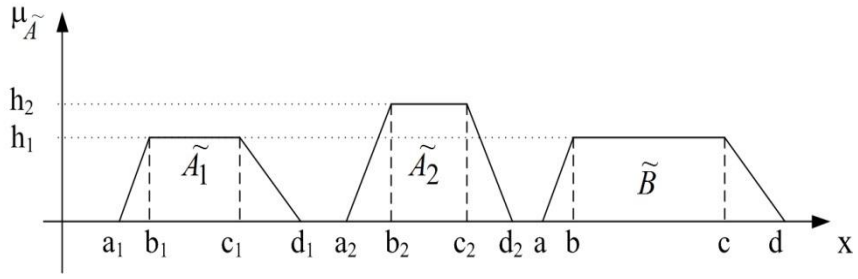

Figure 1. Multiplication of two GTFNs.

In order to prevent an undesirable occurrence (e.g. $b_1<a_1$), the relation (5) must be satisfied in any update of the parameters of $\tilde{A} = (a,b,c,d;h)$ with a quantity $\Delta x \in \{\Delta a, \Delta b, \Delta c, \Delta d\}$. Practically, operations from eq. 4 (Fig. 1) are operations on corresponding intervals (Fig. 2) denoted by: $\overline{X} = [a, b] = [X_L, X_R]$.

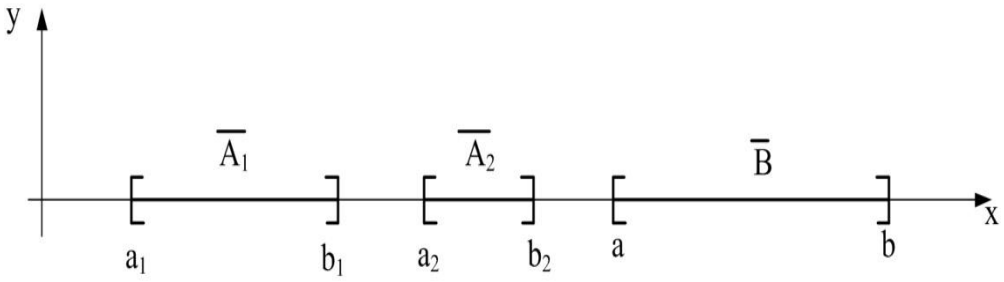

Figure 2. Interval operation involved in multiplication of two GTFNs

The sum between two TGFNs (positive or negative values) using the function principle is given by (1)-(2):

$$\tilde{B} = \tilde{A}_1 \oplus \tilde{A}_2 = (a_1,b_1,c_1,d_1;h_1) \oplus (a_2,b_2,c_2,d_2;h_2) = (a,b,c,d;\min(h_1,h_2)) \quad (6)$$

$$\begin{cases} a = \min(a_1 + a_2, a_1 + d_2, d_1 + a_2, d_1 + d_2) = a_1 + a_2 \\ b = \min(b_1 + b_2, b_1 + c_2, c_1 + b_2, c_1 + c_2) = b_1 + b_2 \\ c = \max(b_1 + b_2, b_1 + c_2, c_1 + b_2, c_1 + c_2) = c_1 + c_2 \\ d = \max(a_1 + a_2, a_1 + d_2, d_1 + a_2, d_1 + d_2) = d_1 + d_2 \end{cases} \quad (7)$$

We can consider the operations described in eq. (2)-(3) as operations on two intervals: [a, d] = [a_1, d_1] ⊗[a_2, d_2] and [b, c] = [b_1, c_1] ⊗[b_2, c_2] with respect to relation (4). This remark will be used in developing a learning algorithm for fully fuzzified neural networks in which all the elements and arithmetic operations are in the GFN framework.

2.2. FUZZY NEURAL NETWOFK WITH GENERALIZED FUZZY NUMBERS (FNN-GFNs)

We suggest the use of the multilayer perceptron (MLP) architecture for FNN-GFNs in a generalized fuzzy number framework (Figure 3). All the variables are TGFNs and all the operations are made according to eq. (1)-(5). Each fuzzy neuron (Fig. 4) has two functions of transfer suggesting a tri-dimensional transfer function based on two functions: $\varphi(\overline{X})$ where $\overline{X}$ are two intervals corresponding to a GTFN that is $\overline{ad}$ and $\overline{bc}$; a function $\psi(h)$ used to transfer the "level" of GTFN.

It is clearly that due to *min* operation between levels of two GTFNs (h_1 and h_2), no further arithmetic operation can produce a value h>min (h_1, h_2) in basic arithmetic operations with other h levels. We introduced the $\psi(h)$ function that can be considered a function that can modify the "energy" level (h) of a GTFN. In equation (9), σ plays the role of threshold from the known formula (8).

$$\varphi(x) = \frac{1}{1 + e^{-x+\theta}} \quad (8)$$

$$\psi(x) = \frac{1}{1 + e^{-x+\sigma}} \quad (9)$$

The architecture of the proposed FNN-GFNs is showed in Fig. 3, the fuzzy neuron is described in Fig. 4 and the activation function for mapping intervals in Fig. 5. This mapping is similar to the approach presented in [5, 6, 15, 16].

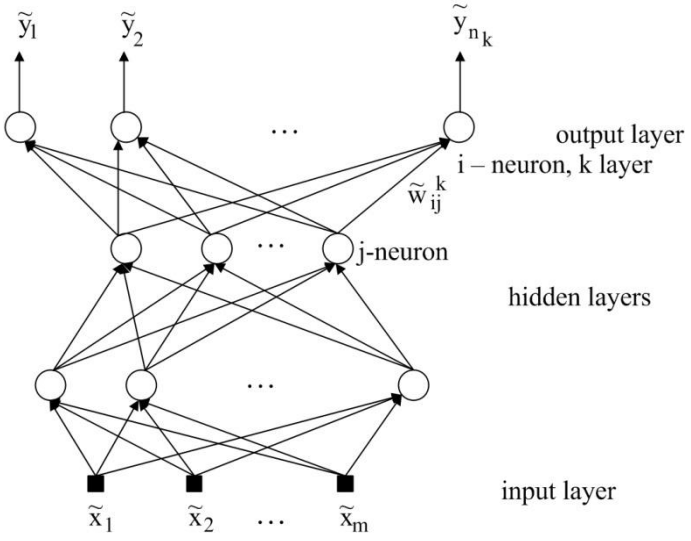

Figure 3. The FNN-GFNs architecture.

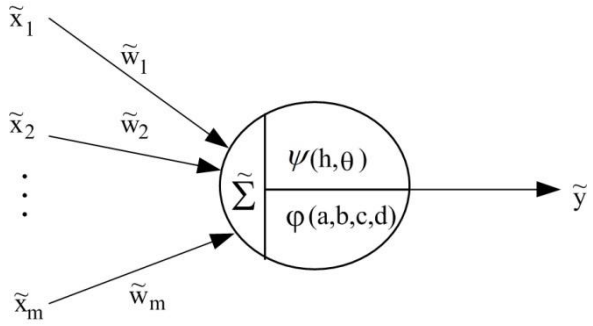

Figure 4. The proposed fuzzy neuron.

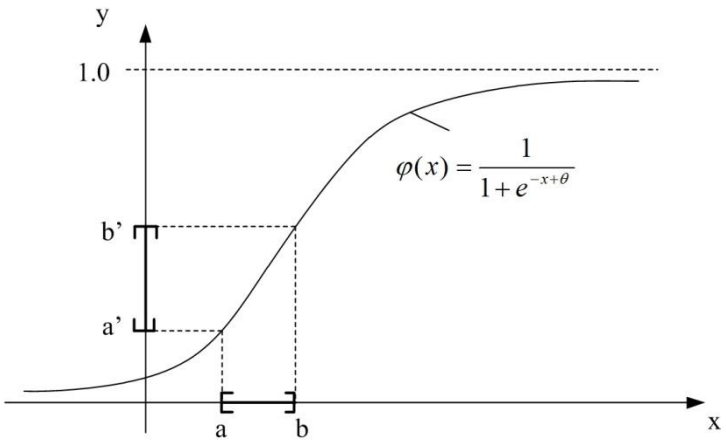

Figure 5. Fuzzy activation function for the fuzzy neuron.

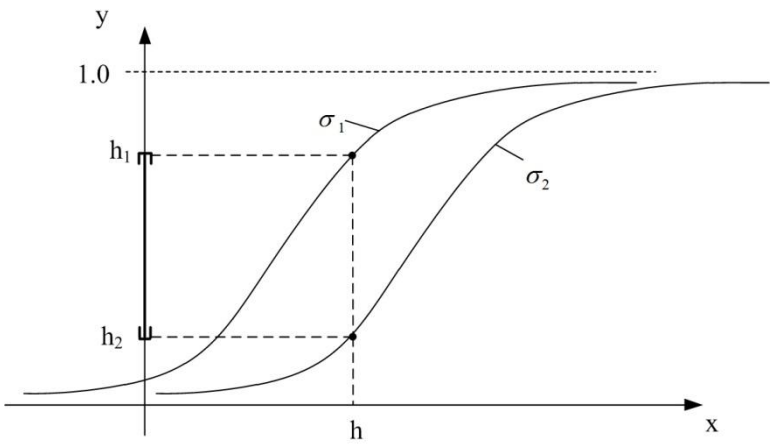

Figure 6. Fuzzy activation function for w values and influence of threshold parameter θ.

All arithmetic operations in FNN-GFNs are made using TGFN and equations (1)-(5) we denote by *y* the output nodes and by *x* the internal or input nodes.

$$\tilde{y}_i = (\varphi(\tilde{s}_i^k), \psi(\tilde{s}_i^k)), \quad \tilde{x}_i^k = (\varphi(\tilde{s}_i^k), \psi(\tilde{s}_i^k)) \tag{10}$$

$$\tilde{s}_i^k = \sum_{j=1}^{n_{k-1}} \tilde{w}_{ij}^k \otimes \tilde{x}_j^{k-1} = \tilde{w}_{i1}^k \otimes \tilde{x}_1^{k-1} \oplus \tilde{w}_{i2}^k \otimes \tilde{x}_2^{k-1} \oplus \ldots \oplus \tilde{w}_{in_{k-1}}^k \otimes \tilde{x}_{n_{k-1}}^{k-1} \tag{11}$$

$$\tilde{s}_i^k = (s_i^{k,a}, s_i^{k,b}, s_i^{k,c}, s_i^{k,d}; h_i^{k,s}) \tag{12}$$

A fuzzy weight $\tilde{w}_{ij}^k$ is a weight that connects the i^{th} fuzzy neuron from layer k with the j^{th} neuron from layer k-1. The input signal that is made by a vector of GTFM propagates simultaneously through each layer of neurons similar to the crisp multilayer perceptron [16].

2.3. THE LEARNING ALGORITHM BASED ON GRADIENT METHOD

In order to develop a learning algorithm for FNN-GFNs, a cost error must be defined. The cost error from crisp case [17] is extended to the fuzzy case [6, 7, 15, 16].

$$E_{total} = 1/2 \cdot \sum_i^n e_i = E_r + E_h = 1/2 \cdot \sum_{i=1, r=a,b,c,d}^{n} (d_i^r - y_i^r)^2 + 1/2 \sum_{i=1}^{n} (d_i^h - y_i^h)^2 \tag{13}$$

Since the h parameter is not related to $\{a, b, c, d\}$ parameters, we can split the E_{total} in two: parts E_r and E_h. If we look at equation (4), (7) we can also split the E_r in two disjoint parts with corresponding intervals: $E_{\overline{ab}}$ and $E_{\overline{cd}}$ that can be treated in a similar manner as relation (5). Let's denote by $p \in \{a, b, c, d, h\}$ a parameter that must be update in order to minimize to total output error E_{total}. Using the gradient method, the parameters p must be updated by:

$$p(t+1) = p(t) + \Delta p(t) \tag{14}$$

$$\Delta p(t) = -\lambda \frac{\partial E_{total}(t)}{\partial p} \tag{15}$$

In order to simplify the formulas, the parameter t (time) is omitted from notations. There are two parameters to be updated: the fuzzy weights and threshold σ. Let's discuss the update of fuzzy weights. As far as the output layer is concerned, we can write (in a similar manner as in [16]). The signal is propagated through feed forward neuronal network, while the error is propagated backwards through the network (back propagation algorithm).

$$\frac{\partial E_{total}}{\partial w_{ij}^{k,a}} = \frac{\partial E_r}{\partial e_{i,r}} \cdot \frac{\partial e_{i,r}}{\partial y_{i,r}} \cdot \frac{\partial y_{i,r}}{\partial s_{i,r}} \cdot \frac{\partial s_{i,r}}{\partial w_{ij}^{k,a}} \tag{16}$$

$$\frac{\partial E_{total}}{\partial w_{ij}^{k,d}} = \frac{\partial E_r}{\partial e_{i,r}} \cdot \frac{\partial e_{i,r}}{\partial y_{i,r}} \cdot \frac{\partial y_{i,r}}{\partial s_{i,r}} \cdot \frac{\partial s_{i,r}}{\partial w_{ij}^{k,d}} \tag{17}$$

Because *min* and *max* operators are not derivable, the last partial derivative in equation (15) is difficult to calculate so we must find a technique that overcomes this difficulty. Few techniques for interval adaptation are presented in [5-7, 15, 16]. A more accurate technique is presented below. Let's denote by $X=[X_L, X_R]$ an interval of real numbers [16], where (X_L, X_R) is one of the pairs (a, d) or (b,c).

$$\rho_{ij}^{k,L}=\frac{\partial s_{i,r}}{\partial w_{ij}^{k,L}}=\frac{\partial\left(\sum_{j=1}^{n_{k-1}}\tilde{w}_{ij}^{k}\otimes\tilde{x}_{j}^{k-1}\right)}{\partial w_{ij}^{k,L}}=\frac{\partial(\min(w_L\cdot x_L,w_L\cdot x_R,w_R\cdot x_L,w_R\cdot x_R))}{\partial w_L}=\ldots \tag{18}$$

$$\frac{\partial(\min(w_L\cdot x_L,w_L\cdot x_R))}{\partial w_L}=\frac{(x_L+x_R)+sign(w_L\cdot x_R)\cdot(x_L-x_R)}{2}$$

$$\rho_{ij}^{k,L}=1/2\cdot[(x_i^{k,L}+x_i^{k,R})+sign(w_L\cdot x_R)\cdot(x_i^{k,L}-x_i^{k,R})] \tag{19}$$

In a similar manner for right value of the interval:

$$\rho_{ij}^{k,R}=\frac{\partial s_{i,r}}{\partial w_{ij}^{k,R}}=\frac{\partial\left(\sum_{j=1}^{n_{k-1}}\tilde{w}_{ij}^{k}\otimes\tilde{x}_{j}^{k-1}\right)}{\partial w_{ij}^{k,R}}=\frac{\partial(\max(w_L\cdot x_L,w_L\cdot x_R,w_R\cdot x_L,w_R\cdot x_R))}{\partial w_R}=\ldots \tag{20}$$

$$\frac{\partial(\max(w_R\cdot x_L,w_R\cdot x_R))}{\partial w_R}=\frac{(x_R+x_L)+sign(w_L\cdot x_R)\cdot(x_R-x_L)}{2}$$

$$\rho_{ij}^{k,R}=1/2\cdot[(x_i^{k,R}+x_i^{k,L})+sign(w_L\cdot x_R)\cdot(x_i^{k,R}-x_i^{k,L})] \tag{21}$$

Finally, if we develop the eq. (15)-(16) we obtain the relations ($q\in\{L,R\}$) taking into account that for the sigmoidal function $\partial\varphi(x)/\partial x=\varphi'(x)=\varphi(x)\cdot[1-\varphi(x)]$.

$$w_{ij}^{k,q}(t+1)=w_{ij}^{k,q}(t)+\lambda\Delta w_{ij}^{k,q}(t)=w_{ij}^{k,q}(t)+\lambda\delta_i^{k,q}\rho_j^{k,q} \tag{22}$$

$$\delta_i^{k,q}=\begin{cases}y_i^{k,q}(1-y_i^{k,q})(d_i^{k,q}-y_i^{k,q}) & \textit{if k is the output layer}\\ x_i^{k,q}(1-x_i^{k,q})\cdot\sum_{j=0}^{n_{k+1}}\delta_j^{k+1,q}\cdot w_{ji}^{k+1,q} & \textit{if k is the input or hidden layer}\end{cases} \tag{23}$$

The next stage is to calculate the adaptation for second transfer function $\psi(x, \sigma)$. The same method as in *back propagation* is applied here. The learning coefficient $\lambda\in(0, 1)$ has the same meaning as in [16]. Sometimes, $\lambda>1$ if the convergence of the algorithm is too slow.

$$\frac{\partial E_{total}}{\partial\sigma_i^k}=\frac{\partial E_h}{\partial e_i}\cdot\frac{\partial e_i}{\partial y_i}\cdot\frac{\partial\psi(h_i^k,\sigma_i^k)}{\partial\sigma_i^k}=(d_i^h-y_i^h)y_i^h(1-y_i^h)\cdot\frac{\partial y_i}{\partial\sigma_i^k}=-(d_i^h-y_i^h)y_i^h(1-y_i^h)h_i^k \tag{24}$$

$$h_i^k = \min\{\bigcup_j h_j^k\},\ j \in \{w_{ij}^{k,r}, x_i^{k,r}\} \tag{25}$$

For input hidden layers, the formula above becomes as in (22) with missing *w* term. However, even the adaptive solution proposed above has developed in order to have balanced values for σ in all the fuzzy neurons, in all the layers an unbalanced exact solution can be developed.

Let's set all the σ values to 0.5 e.g. for all the neurons in all the layers except the output layer. For each neuron in the output layer we will have an input $h_i \in (0, 1]$ value in the normalized case. Let's suppose that we have to obtain an output value $d_i^h \in (0,1)$, where the limit of interval to value "1" is approximated to a given tolerance. Then, σ_i is given by:

$$\sigma_i = h_i + \ln\left(\frac{1-d_i^h}{d_i^h}\right) \tag{26}$$

The algorithm can be summarized as follows:

Step 1. Initialize the fuzzy weights with random GTFN. Set all the *h* values of weights and neurons to 0.5.

Step 2. Feed forward input signal to output.

Step 3. Estimate $\Delta w_{ij}^{k,q}$ by back propagation of error according to formulas (21)-(22)

Step 4. Update each weight only if condition (5) is satisfied. If this condition is not satisfied, the weight remains unchanged.

Step 5. Compute the total error E_r, eq. (13). If E_r is smaller than a desired value or the number of iterations has reached a prescribed number, go to Step. 6, otherwise go to Step 2.

Step 6. Compute the exact values for hi in the output layer.

Step 7. Stop the algorithm

3. RESULTS AND DISCUSSION

The proposed FNN with GTFN and the proposed algorithm has been tested in mapping a vector of GTFN into another vector of GTFN. We have chosen architecture with 4 inputs, 5 neurons in hidden layer and 3 neurons in output layer (targets). The learning factor λ=0.54 has been chosen experimentally. The maximum iteration number was set to 250. The results of good performance are showed in Table 1.

Table 1. Input/Output mapping of two vectors of GTFN.

Inputs	(-0.2,-0.1, 0.1, 0.3)	(0.2, 0.3, 0.1, 0.4)	(-0.2,0.4, 0.7, 0.7)	(-0.5,-0.1, 0.7, 0.9)
Target	(0.2, 0.3, 0.5, 0.9)	(0.2, 0.3, 0.5, 0.3)	(0.2, 0.3, 0.5, 0.8)	
Output	(0.1998, 0.3002, 0.502, 0.9)	(0.2015, 0.300, 0.5001, 0.3)	(0.2000, 0.3001, 0.4999, 0.8)	

The proposed algorithm is applicable not only to positive GTFN but also to negative values of the tuple $\{a, b, c, d\}$ (one or more values in the tuple). Another solution to deal with negative values is one inspired from crisp neural networks. We can map (or translate) the input interval into a convenient interval for all the values and, after training, the values are re-mapped into initial interval. The same line of thinking is applied to output values.

The function ψ can also have other forms, the activation function for crisp fuzzy neuron. A particular interest is a linear limited one [16]. This is much more suitable for calculating σ_i values, especially for prediction usage of FNN. In this case, however, the computation by direct calculation of σ_i is not suitable and an adaptive form is necessary to make a tradeoff among desired h_i values of output during the trajectory of desired vectors of GTFN.

4. CONCLUSION

Fuzzification of multilayer perceptron is proposed using TGFN and an extension for activation function of fuzzy neurons. The proposed method is proven to be feasible in an example of mapping input-output vectors of GTFN of different dimensions. The results are very good, the maximum total error is below $0.4 \cdot 10^{-2}$ in the worst-case scenario. A learning algorithm for $\varphi(x,\theta)$ will be developed in further research. The new parameter θ for each fuzzy neuron will be adapted using a gradient method. It is supposed that the new parameter can improve the accuracy of the proposed algorithm

REFERENCES

[1] Dubois, D., Prade, H., *Int. J. Systems Sci*., **9,** 613, 1978.
[2] Kaufmann, A, Gupta, M.M., *Introduction to fuzzy Arithmetic Theory and Applications*, Van Nostrand Reinhold, New York, 1991.
[3] Buckley, J.J., Hayashi, Y., *Fuzzy Sets and Systems*, **66,** 1, 1994.
[4] Hayashi, Y., Buckley, J.J., Czogala E., *Int. J. Intell. Systems*, **8**, 527, 1993.
[5] Ishibuchi, H., Morioka, K.,Turksen, I.B., *International Journal of Approximate Reasoning*, **13**(4), 327, 1995.
[6] Ishibuchi, H., Kwon, K., Tanaka, H., *Fuzzy Sets and Systems*, **71**(3), 277, 1995 .
[7] Ishibuchi, H., Morioka, K., Turksen, I.B., *International Journal of Approximate Reasoning*, **13**(4), 327, 1995.
[8] Wua, W., Wanga, J., Chenga, M., Lia, Z., *Neural Networks*, **24** (1), 91, 2011.
[9] Buckley, J.J., Hayashi, Y., Fuzzy *Sets and Systems*, **61**, 43, 1993.
[10] Liu, P., Fuzzy *Sets and Systems*, **119**, 313, 2001.

[11] Chen, S.H, Tamkang *Journal of Management Sciences*, **6**(1), 13, 1985.
[12] Chen, S.H., Hseih, C.H., Journal *of the Chinese Fuzzy System Association*, **5**(2), 1, 2000.
[13] Ebrahimnejad, A., Applied *Soft Computing*, **19**, 171, 2014.
[14] Karthik, J.N., Chandrasekaran, E., *International Journal of Computer Applications*, **64**(9), 35, 2013.
[15] Arotaritei, D., Teodorescu, H.N., *Proceedings of the 5th European Congress on Intelligent Techniques and Soft Computing EUFIT'97*, **1,** 261, 1997.
[16] Teodorescu, H.N., Arotaritei, D.,Gonzales, E.P., Cuervo, G.M., *Proceedings of the International Conference on Intelligent Technologies in Human-Related Sciences including the SYS'96 Symposium on Systems and Signals ITHURS' 96*, **1**, 81, 1996.