Proyecciones Journal of Mathematics
Vol. 28, N^o 3, pp. 239–252, December 2009.
Universidad Católica del Norte
Antofagasta - Chile
DOI: 10.4067/S0716-09172009000300005

FUZZY GRAPHS IN FUZZY NEURAL NETWORKS

K. SAMEENA
and
M. S. SUNITHA
NATIONAL INSTITUTE OF TECHNOLOGY, INDIA
Received : May 2009. Accepted : October 2009

Abstract

In this paper we observe that, the fuzzy neural network architecture is isomorphic to the fuzzy graph model and the output of a fuzzy neural network with OR fuzzy neuron is equal to the strength of strongest path between the input layer (particular input neuron/neurons) and the out put layer(particular output neuron). We explain this result through an example, which describes the marketability of text books of kindergarten classes.

AMS 2000 Subject Classification : 04A72 , 05cxx

Keywords *: Fuzzy graph,strongest path, fuzzy neuron, OR fuzzy neuron.*

1. Introduction

A fuzzy graph [1] is a pair $G : (\sigma, \mu)$ where σ is a fuzzy subset of a set S and μ is a fuzzy relation on σ such that $\mu(u, v) \leq \sigma(u) \bigwedge \sigma(v)$. We assume that S is finite and nonempty, μ is reflexive and symmetric[1]. Also, we denote the underlying crisp graph by $G^* : (\sigma^*, \mu^*)$ where $\sigma^* = \{u \epsilon S : \sigma(u) > 0\}$ and $\mu^* = \{(u, v) \epsilon SXS : \mu(u, v) > 0\}$. A path P of length n is a sequence of distinct nodes $u_0, u_1, ...u_n$ such that $\mu(u_{i-1}, u_i) > 0, i = 1, 2, ..., n$ and the degree of membership of a weakest arc is defined as its strength. The strength of connectedness between two nodes x and y is defined as the maximum of the strengths of all paths between x and y and is denoted by $CONN_G(x, y)$. An $x - y$ path P is called a strongest $x - y$ path if its strength equals $CONN_G(x, y)$[1]. A fuzzy graph $G : (\sigma, \mu)$ is connected if for every x,y in $\sigma^*, CONN_G(x, y) > 0$.

Neural networks are simplified models of the biological nervous system and therefore have drawn their motivation from the kind of computing performed by a human brain. Neural networks exhibit characteristic such as mapping capabilities or pattern association, generalization, robustness, fault tolerance, and parallel and high speed information processing. Fuzzy neural networks and neural fuzzy systems are powerful techniques for various computational and control applications. The area is still under a great influx from both theoretical and applied research. There is no systematic or unified approach for incorporating the concepts of fuzziness and neural processing. Fuzzy sets can be used to describe various aspects of Neural Computing. That is, fuzziness may be introduced at the input out put signals, synaptic weights, aggregation operation and activation function of individual neurons to make it fuzzy neuron. Different aggregation operations and activation functions result in fuzzy neurons with different properties. Thus there are many possibilities for fuzzification of an artificial neuron. So we may find a variety of fuzzy neurons in the literature [2], [3], [4] and [5].

Applying fuzzy methods into the workings of neural networks constitutes a major thrust of neuro-fuzzy computing. A fuzzy neuron has the same basic structure as the artificial neuron except that its components and parameters are described through the mathematics of fuzzy logic. There are many possibilities for fuzzification of an artificial neuron so we may find a variety of fuzzy neurons in the literature. A fuzzy neuron consists of external inputs, synapses/synaptic weights, dendrites, soma, and an axon

through which the neural output is transmitted to other neurons. The external inputs $x_1, x_2, .., x_n$ enter the j^{th} neuron and gets modified by the synaptic weights $w_{ji}, w_{j2}, ..., w_{jn}$. Each synaptic output forms an input to the processing element (soma), called the dendritic input. The external input $x_1, x_2, ., x_n$ are of fuzzy signals bounded by membership values over the interval $[0, 1]$. The synaptic weights w_{ji} are also defined over the interval $[0, 1]$.

Figure shows a fuzzy neuron:

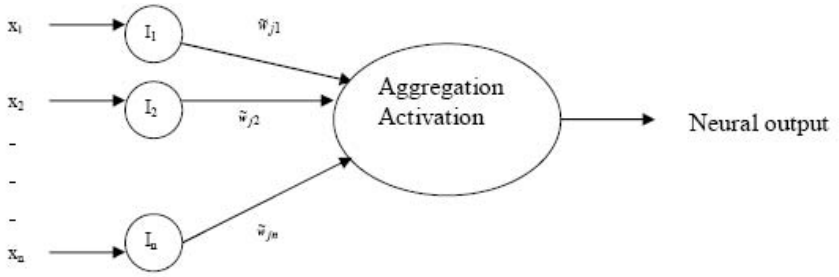

Figure 1 : A Fuzzy Neuron

2. AND and OR fuzzy neurons:

AND and OR fuzzy neurons are special class of fuzzy neurons. They are shown in below,

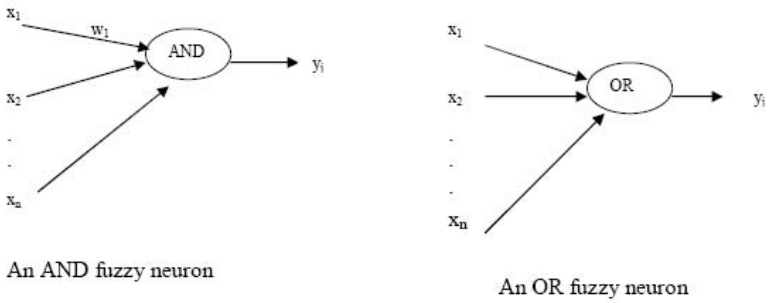

Figure 2 : AND and OR fuzzy neurons

These neurons employ Max and Min operations for forming dendritic inputs to the neuron (soma) and for aggregating them.
The OR fuzzy neuron first perform an AND operation for Input modification between external input x_i and corresponding weight w_{ji}. Subsequently it uses an OR operation to perform the aggregation of the dendritic inputs. There fore the output of an OR fuzzy neuron can be written as,

$y_j = \bigvee_{i=1}^{n}(x_i \wedge w_{ji})$

The AND neuron, on the other hand, performs a complementary computation. There fore the output of an AND fuzzy neuron is,

$y_j = \bigwedge_{i=1}^{n}(x_i \vee w_{ji})$

3. Fuzzy BP Architecture:

Fuzzy BP is a three layered feed forward architecture [4]. The three layers are input layer, hidden layer and output layer. Figure 3 illustrates the architecture of fuzzy BP.

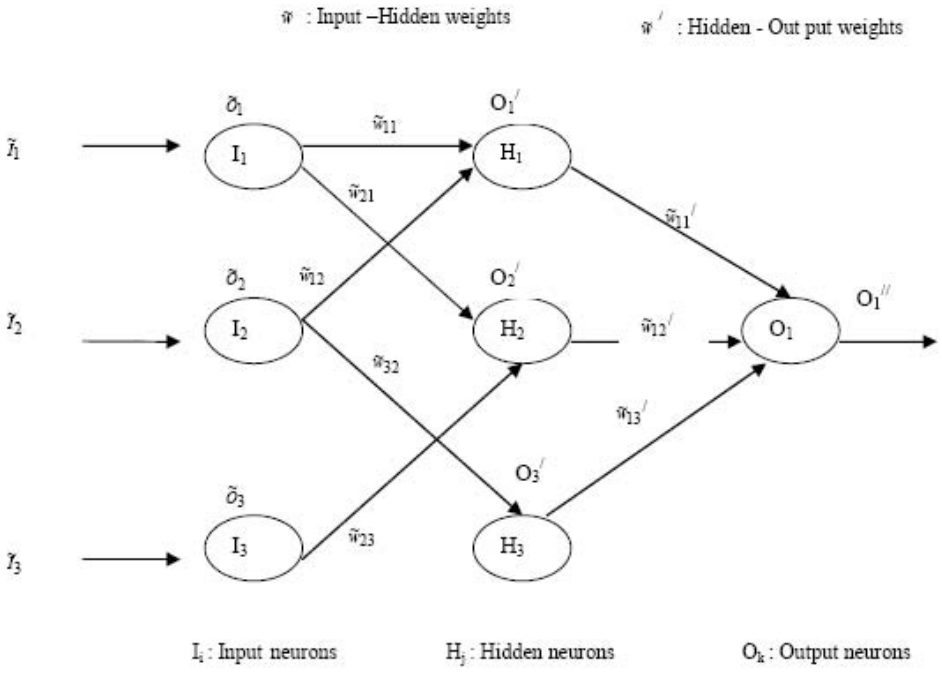

Figure 3 : A Fuzzy BP Architechture

Here $\tilde{I}_1 = (\tilde{I}_1, \tilde{I}_2, \tilde{I}_3)$ indicates the first input component of the input pattern, $\tilde{O}_i$ be the out put value of the i^{th} input neuron $O_j^{/}$ and $O_k^{//}$ are the j^{th} and k^{th} outputs of the hidden and output layer neurons respectively. $\tilde{w}_{ji}$ and $\tilde{w}'_{kj}$ are the fuzzy connection weights between the i^{th} input neuron and j^{th} hidden neuron , and the j^{th} hidden neuron and k^{th} output neuron respectively. In addition $\wedge$ is the aggregation function and $\vee$ is the activation function.
The computation carried out by each layer is as follows [4],
Input neurons

$$\tilde{O}_i = \tilde{I}_i \quad , i = 1, 2, 3 \tag{3.1}$$

Hidden neurons,

$$NET_j = \wedge(\tilde{w_{ji}}, \tilde{O}_i) \quad j, i = 1, 2, 3. \tag{3.2}$$

$$O'_j = \vee(NET_j) \tag{3.3}$$

$$NET'_j = \wedge(\tilde{w_{kj}}', O'_j) \tag{3.4}$$

Out put neurons,

$$O''_k = \vee(NET'_j), \quad k = 1 \tag{3.5}$$

The learning procedure of fuzzy BP follows the gradient descent method of minimizing error E due to the learning . During the learning phase the weights are adjusted so as to minimize E. The weight change at time t is given by, $\Delta\tilde{w}(t) = -\eta\nabla E(t) + \alpha\Delta\tilde{w}(t-1)$ where η is the learning rate and α is a constant value. $\alpha\Delta\tilde{w}(t)$ is the momentum term to be added for speeding up convergence [4].

4. Fuzzy graph model for OR-fuzzy neuron

An artificial neural network can be represented by a directed graph. The vertices of the graph may represent neurons (input/ output) and the edges are labeled by the weights attached to the synaptic links. Now let S be a non empty set. In a fuzzy neural network we take each neuron as a node with unit weight and the relationships (connections) between the neurons

(only forward direction) as arcs with weight (specifying the strength of connectivity) in the interval [0,1].
Now consider the example,

Example 1 : Draw a feed forward network for a 1-2-1 architecture with input-hidden weight matrix $\begin{pmatrix} 0.2 \\ 0.5 \end{pmatrix}$ and hidden- output weight matrix $\begin{pmatrix} 0.4 \\ 0.5 \end{pmatrix}$.
We can draw the neural network architecture as in Figure 4.

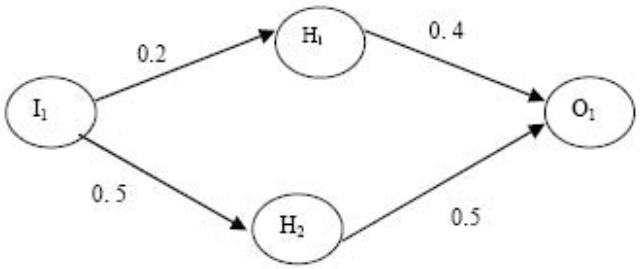

Figure 4 : A Fuzzy Neural Net Work

Now we can draw the corresponding fuzzy graph model of example 1 as $G : (\sigma, \mu)$ with $\sigma^* = \{I_1, H_1, H_2, O_1\}$ with $\mu(I_1, H_1) = 0.2, \mu(I_1, H_2) = 0.5, \mu(H_1, O_1) = 0.4$ and $\mu(H_2, O_1) = 0.5$. The fuzzy graph model is also same as Figure 4.

So we say that fuzzy neural network architecture is isomorphic to a fuzzy graph model. We take the OR fuzzy neuron in the fuzzy neural network model and we observe that the out put of an OR fuzzy neuron is the strength of the strongest path (paths) between the input and the out put neurons and that particular strongest path (paths) are identified.

It follows from the equations (3.1) to (3.5) that the fuzzy neural out put with OR fuzzy neuron is nothing but the strength of connectivity between an input node (nodes) and an out put node.

We make use of the inputs presented here to demonstrate the application of fuzzy graph model using OR fuzzy neurons . The class of fuzzy reasoning problems involve multi input - output systems where generally a single fuzzy set describes the output variable. This methodology is employed when there are several factors acting as inputs directly or indirectly affecting the output of a typical system. In this case, all effects of inputs are to be aggregated to find the out put fuzzy set. To get the desired output, we may train the input - output data using fuzzy neural network. Consider the example below :

Example 2: Marketability of a text book of Kindergarten class:

The marketability of text books of Kindergarten classes can be studied based on the following fuzzy rules of implications:

- If examples are MORE, then better SALE
- If cost is LOW, then better SALE
- If figures are MORE, then better SALE

Now, suppose that by "better SALE " , we mean a sale of 60 percentage of books out of the total stock and we may fix our target output of 0.6 for the output variable ,y - better SALE. Next, consider an input pattern of $[0.63/x_1, 0.02/x_2, 0.6/x_3]$, where the input variables are
x_1 – Examples are MORE
x_2 – Cost is LOW
x_3 – Figures are MORE

Note that x_1 is given the membership value 0.63, which can be interpreted as, about 63% of the pages of the book contain examples. Similarly, x_2 has membership value 0.02, which indicates that the cost is not that LOW. Finally , x_3 has given a membership value 0.6, which shows that, about 60 % of the pages of the book contains figures.

Using a 3-3-1 architecture [i.e., a three layer fuzzy neural network with three input nodes in the input layer, one hidden layer with three nodes and one output node in the output layer], we train the data by fuzzy BP(back propagation) algorithm with the input-hidden weight matrix as

$\begin{pmatrix} 0.5 & 0.5 & 0.0 \\ 0.5 & 0.0 & 0.5 \\ 0.0 & 0.5 & 0.0 \end{pmatrix}$ and the hidden output weight matrix as $\begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \end{pmatrix}$.

Training the Network:

Iteration 1 ,
$\tilde{I} = (0.63, 0.02, 0.6)$. The output target D = 0.6.
The output of the input neurons $\tilde{O_i}$ = (0.63, 0.02,0.6), by equation (3.1)
The input to the hidden neurons are obtained as follows, using equation (3.2),
$NET_1 = (\wedge(0.63, 0.5), \wedge(0.02, 0.02)) = (0.5, 0.02)$
$NET_2 = (\wedge(0.63, 0.5), \wedge(0.6, 0.5)) = (0.5, 0.5)$
$NET_3 = \wedge(0.02, 0.02) = 0.02.$
The output of the hidden neurons is, (using equation(3.3))

$O_1^{'} = \vee(0.5, 0.02) = 0.5$
$O_2^{'} = \vee(0.5, 0.5) = 0.5$
$O_3^{'} = 0.02$

Now the output of the output neuron (using equation (3.4) and equation (3.5)),

$NET_1' = \wedge(0.5, 0.5) = 0.5$
$NET_2' = \wedge(0.5, 0.5) = 0.5$
$NET_3' = \wedge(0.02, 0.02) = 0.02$

$O_1'' = \vee(0.5, 0.5, 0.02) = 0.5$

The modified network is shown below:

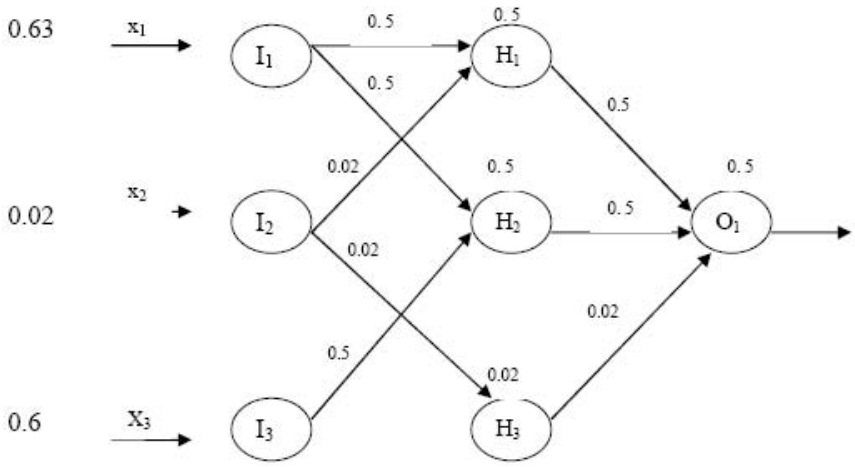

Figure 5 : The modified network

Here the out put is 0.5, which is the strength of the strongest path from the inputs x_1(Examples are MORE) and x_3 (Figures are MORE) to the output(Better SALE). The strongest paths are, $(I_1)-(H_1)-(O_1)$, $(I_1)-(H_2)-(O_1)$ and $(I_3)-(H_2)-(O_1)$.

We now proceed to compute the change of weights $\Delta\tilde{w}(t)$ and $\Delta\tilde{w}'$ for the input-hidden and hidden-output layers respectively [4].

Initially set $\Delta\tilde{w}(t-1)=0$, choose η= 0.9 and $\alpha=0.1$, arbitrarily.
Also $\tilde{w}(t-1)=0$ and $\tilde{w}'(t-1)=0$.
Now compute change of weights $\Delta\tilde{w}'(t)$ for hidden - output layer as follows [4].

Compute $\nabla E_p(t)=\frac{\partial E_p}{\partial w'_{pj}}$. Here $p=1$ and $j=1,2,3$, since $p=1$, we replace E_p by E in the following equations.

$$\frac{\partial E}{\partial w'_{1j}}=-(D-O''_1)\ O''_1\ (1-O''_1)\ O'_j,\quad j=1,2,3.$$

$\frac{\partial E}{\partial w'_{11}} = -(D - O''_1)\ O''_1\ (1 - O''^1)O'_1 = -(0.6 - 0.5)0.5(1 - 0.5)0.5 = -0.0125$

$\frac{\partial E}{\partial w'_{12}} = -(D-O''_1)O''_1(1-O''_1)O'_2 = -(0.6-0.5)0.5(1-0.5)0.5 = -0.0125$

$\frac{\partial E}{\partial w'_{13}} = -(D - O''_1)O''_1(1 - O''_1)O'_3 = -(0.6 - 0.5)0.5(1 - 0.5)0.02 = -0.0005$

Next compute $\Delta\tilde{w}'_{kj}(t) = -\eta\nabla E(t) + \alpha\Delta\tilde{w}'(t-1)$

$\Delta\tilde{w}'_{11}(t) = -0.9 \times -0.0125 + 0.1 \times 0 = 0.01125,$

$\Delta\tilde{w}'_{12}(t) = -0.9 \times -0.0125 + 0.1 \times 0 = 0.01125$ and

$\Delta\tilde{w}'_{13}(t) = -0.9 \times -0.0005 + 0.1 \times 0 = 0.00045.$

Update the weight for hidden -output layer as ,

$\tilde{w}'_{kj}(t) = \tilde{w}'_{kj}(t-1) + \Delta\tilde{w}'_{kj}(t), \quad j = 1, 2, 3$ and $k = 1.$

$\tilde{w}'_{11}(t) = \tilde{w}'_{11}(t-1) + \Delta\tilde{w}'_{11}(t) = 0.5 + 0.01125 = 0.51125$

$\tilde{w}'_{12}(t) = \tilde{w}'_{12}(t-1) + \Delta\tilde{w}'_{12}(t) = 0.5 + 0.01125 = 0.51125$ and

$\tilde{w}'_{13}(t) = \tilde{w}'_{13}(t-1) + \Delta\tilde{w}'_{13}(t) = 0.02 + 0.00045 = 0.02045.$

Next compute change of weights for the input -hidden layer as follows.

Let $\delta = -(D - O"_1)O"_1(1 - O"_1) = -(0.6 - 0.5)0.5(1 - 0.5) = -0.025$

Also $\nabla E(t) = \frac{\partial E}{\partial w_{ji}} = (\Sigma_{k=1}\delta\tilde{w}'_{kj})O'_j(1 - O'_j)\tilde{O}_i, \quad k = 1, \quad j = 1, 2, 3$ and $i = 1, 2, 3.$

That is $\nabla E(t) = \frac{\partial E}{\partial w_{ji}} = (\delta\tilde{w}'_{kj})O'_j(1-O'_j)\tilde{O}_i, \ \ j = 1, 2, 3$ and $i = 1, 2, 3.$

$\frac{\partial E}{\partial w_{11}} = (\delta\tilde{w}'_{11})O'_1(1 - O'_1)\tilde{O}_1 = (-0.025 \times 0.51125)0.5(1 - 0.5)0.63 = -0.002013$

$\frac{\partial E}{\partial w_{21}} = (\delta\tilde{w}'_{12})O'_1(1-O'_1)\tilde{O}_1 = (-0.025 \times 0.51125)0.5(1-0.5)0.63 = -0.002013$

$\frac{\partial E}{\partial w_{12}} = (\delta\tilde{w}'_{12})O'_2(1-O'_2)\tilde{O}_2 = (-0.025 \times 0.51125)0.5(1-0.5)0.02 = -0.000064.$

$\frac{\partial E}{\partial w_{32}} = (\delta\tilde{w}'_{13})O'_3(1-O'_3)\tilde{O}_2 = (-0.025 \times 0.02045)0.02(1-0.02)0.02 = -0.00000002.$

$\frac{\partial E}{\partial w_{23}} = (\delta\tilde{w}'_{12})O'_2(1-O'_2)\tilde{O}_3 = (-0.025 \times 0.51125)0.5(1-0.5)0.6 = -0.001917.$

Next compute $\Delta\tilde{w_{ji}}(t) = -\eta\nabla E(t) + \alpha\Delta\tilde{w}_{ji}(t-1)$

$\Delta\tilde{w}_{11}(t) = -\eta\nabla E(t) + \alpha\Delta\tilde{w}_{11}(t-1) = -0.9 \times -0.002013 + 0.1 \times 0 = 0.001812$

$\Delta\tilde{w}_{21}(t) = -\eta\nabla E(t) + \alpha\Delta\tilde{w}_{21}(t-1) = -0.9 \times -0.002013 + 0.1 \times 0 = 0.001812$

$\Delta\tilde{w}_{12}(t) = -\eta\nabla E(t) + \alpha\Delta\tilde{w}_{12}(t-1) = -0.9 \times -0.000064 + 0.1 \times 0 = 0.0000576$

$\Delta\tilde{w}_{32}(t) = -\eta\nabla E(t) + \alpha\Delta\tilde{w}_{32}(t-1) = -0.9 \times -0.00000002 + 0.1 \times 0 = 0.000000018$

$\Delta\tilde{w}_{23}(t) = -\eta\nabla E(t) + \alpha\Delta\tilde{w}_{23}(t-1) = -0.9 \times -0.001917 + 0.1 \times 0 = 0.0017253$

Modified weight for input hidden layer;

$\tilde{w_{ji}} = \tilde{w}_{ji}(t-1) + \Delta\tilde{w}_{ji}(t-1)$

$\tilde{w}_{11} = \tilde{w}_{11}(t-1) + \Delta\tilde{w}_{11}(t-1) = 0.5 + 0.001812 = 0.501812$

$\tilde{w}_{21} = \tilde{w}_{21}(t-1) + \Delta\tilde{w}_{21}(t-1) = 0.5 + 0.001812 = 0.501812$

$\tilde{w}_{32} = \tilde{w}_{32}(t-1) + \Delta\tilde{w}_{32}(t-1) = 0.02 + 0.000000018 = 0.020000018$

$$\tilde{w}_{12} = \tilde{w_{12}}(t-1) + \Delta\tilde{w}_{12}(t-1) = 0.02 + 0.0000576 = 0.0200576$$

$$\tilde{w_{23}} = \tilde{w}_{23}(t-1) + \Delta\tilde{w}_{23}(t-1) = 0.5 + 0.0017253 = 0.5017253$$

Using OR fuzzy neuron we have,
By equation (3.2)
$NET_1 = [\wedge(0.501812, 0.63), \wedge(0.0200576, 0.02)] = (0.501812, 0.02)$
$NET_2 = [\wedge(0.501812, 0.63), \wedge(0.5017253, 0.6)] = (0.501812, 0.5017253)$
$NET_{3.3} = \wedge(0.020000018, 0.02) = 0.02$
$O_1^{'} = 0.501812 \quad O_2^{'} = 0.501812 \quad O_3^{'} = 0.02$ by equation (3.3)
By equation (3.4)
$NET_1^{'} = \wedge(0.501812, 0.51125) = 0.501812, NET_2^{'} = \wedge(0.501812, 0.51125) = 0.501812, NET_3^{'} = \wedge(0.02, 0.02045) = 0.02$
$O_1^{''} = \vee(0.501812, 0.501812, 0.02) = 0.501812$ by equation (3.5)
At the end of first iteration after training the fuzzy BP model with one input pattern, the updated neural network architecture is shown below:

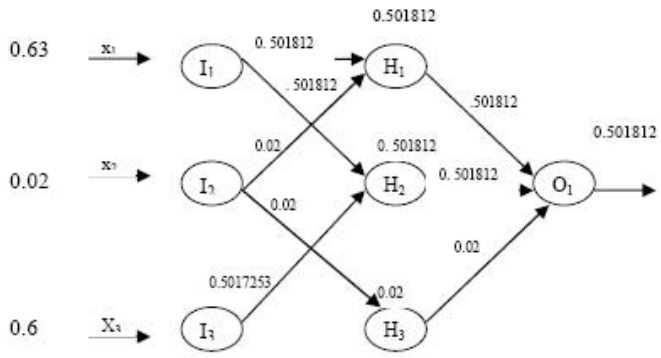

Figure 6 : The modified network after training

Note that the output is 0.501812, which is the strength of the strongest path between (I_1) and (O_1) and the strongest paths are $(I_1) - (H_1) - (O_1)$ and $(I_1) - (H_2) - (O_1)$. That is the output (better SALE) is due to the input(examples are MORE).

From the above illustration it follows that for 'better SALE' of the textbooks 'examples should be MORE'. Note that this observation is after just one training of the neural network.

This process is repeated over and over until the error factor becomes a small value and at this point the desired out put and actual out put is approximately same. How ever, it is important to note that the network should be trained with sufficient samples (and sufficient number of times), to obtain desired results.

5. Conclusion

Until now the power of fuzzy graphs has not exploited in fuzzy neural networks. Also different fuzzy neural networks have different underlying topologies, so it is more convenient and flexible to utilize fuzzy graph theory when analyzing and designing fuzzy neural networks. Once the fuzzy graph model for a specific problem is obtained, we can directly go for the corresponding topology of fuzzy neural network. So we can say that the fuzzy neural network architecture is isomorphic to the fuzzy graph model. Also we observe that the output of a fuzzy neural network with OR fuzzy neuron is equal to the strength of strongest path between the input layer (particular input neuron/neurons) and the out put layer(particular output neuron). We explain this result through an example, which describes the marketability of text books of kindergarten classes.

References

[1] Mordeson. J. N, Nair. P. S, Fuzzy Graphs and Fuzzy Hyper Graphs, Physica-Verlag, (2000).

[2] A. M Ibrahim, Introduction to Applied Electronics, Prentice Hall Of India, (1999).

[3] L. H Tsoukalas, R. E Uhrig, fuzzy and Neural Approaches in Engineering, John Wiley & Sons, (1997).

[4] S. Rajesekaren, G. A Vijayalakshmi Pai, Neural Networks, Fuzzy Logic and Genetic Algorithms, Synthesis and Applications, Prentice- Hall of India, (2003).

[5] Timothy J Ross , Fuzzy Logic with Engineering Application, Mc Graw-Hill, (1997).

[6] Jang. J. S. R., Sun. C. T, Mizutani. E, Neuro - Fuzzy and soft computing, Prentice - Hall upper saddle River, N J, (1997).

Sameena K.
Department of Mathematics
National Institute of Technology
Calicut - 673 601
India
e-mail : sameenakalathodi@gmail.com

and

M. S. Sunitha
Department of Mathematics
National Institute of Technology
Calicut - 673 601
India
e-mail : sunitha@nitc.ac.in