ASSIGNMENT - 7

AIM : Implementation of a simple Neural Network for backward and forward propagation (without tool).

OBJECTIVE :

- To understand the concept of backward propagation.
- To understand the concept of forward propagation.
- To implement simple NN(Neural Network) for backward and forward propagation.

SOFTWARE REQUIREMENTS :

- Linux Operating System
- Java Compiler
- Eclipse IDE

MATHEMATICAL MODEL :

Consider a following set theory notations related to a program. The mathematical model M for Neural Network is given as below,

M={S,So,A,G}

Where,
S=State space i.e.All the possible intermediate states
So= Initial State i.e Input matrix of size a[4][2] ,output matrix of size a[4], learning rate, randomly generated weights(w1,w2).
A=Set of Actions/Operators i.e A : S->S' Here, it moves forward and backward.
G=Goal state.i.e Number of epochs.

THEORY :

NEURAL NETWORK:

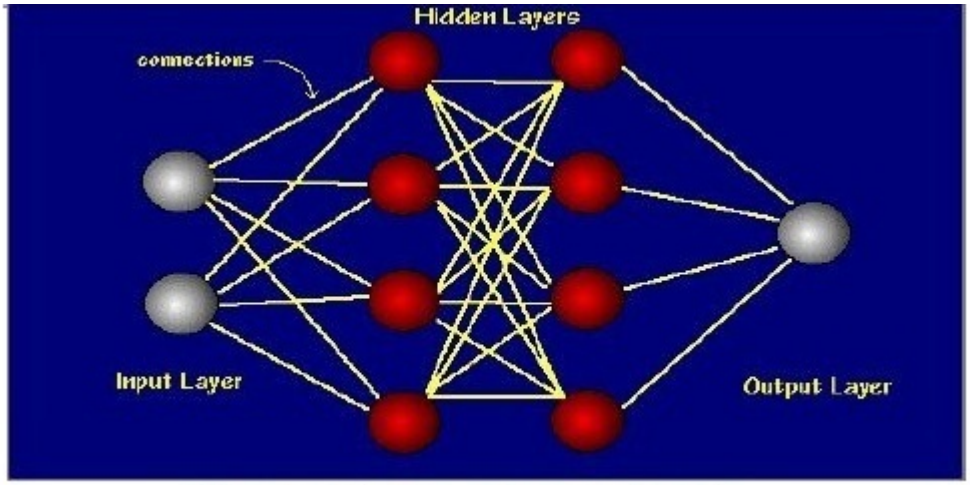

Fig : Architecture of neural network

A neural network is defined as, "A computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs."

Neural neworks are typically organized in layers. Layers are made up of a number of interconnected 'nodes' which contain an 'activation function'. Patterns are presented to the network via the 'input layer', which communicates to one or more 'hidden layers' where the actual processing is done via a system of weighted 'connections'. The hidden layers then link to an 'output layer' where the answer is output as shown in the above figure.

BACKWARD PROPAGATION :

Backpropagation is a supervised learning algorithm and is mainly used by Multi-Layer-Perceptrons to change the weights connected to the net's hidden neuron layer(s).The backpropagation algorithm uses a computed output error to change the weight values in backward direction. To get this net error, a forward propagation phase must have been done before. While propagating in forward direction, the neurons are being activated using the sigmoid activation function.

The formula of sigmoid activation is:

$$\mathbf{f(x) = \frac{1}{1+e^{-input}}}$$

The algorithm for backward propagation works as follows:

1. Perform the forwardpropagation phase for an input pattern and calculate the output error.

2. Change all weight values of each weight matrix using the formula weight(old) + learning rate * output error * output(neurons i) * output(neurons i+1) * (1 - output(neurons i+1)) .

3. Go to step 1

4. The algorithm ends, if all output patterns match their target patterns

Backward Propapagtion : Step-by-Step Example

Suppose you have the following 3-layered Multi-Layer-Perceptron:
Patterns to be learned:

Input	**Target**
0 $\bar{1}$	0
1 $\bar{1}$	1

First, the weight values are set to random values: 0.62, 0.42, 0.55, -0.17 for weight matrix 1 and 0.35, 0.81 for weight matrix 2.

The learning rate of the net is set to 0.25.

Next, the values of the first input pattern (0 1) are set to the neurons of the input layer (the output of the input layer is the same as its input).

The neurons in the hidden layer are activated:

Input of hidden neuron 1: 0 * 0.62 + 1 * 0.55 = 0.55
Input of hidden neuron 2: 0 * 0.42 + 1 * (-0.17) = -0.17
Output of hidden neuron 1: 1 / (1 + exp(-0.55)) = 0.634135591
Output of hidden neuron 2: 1 / (1 + exp(+0.17)) = 0.457602059

The neurons in the output layer are activated:

Input of output neuron: 0.634135591 * 0.35 + 0.457602059 * 0.81 = 0.592605124
Output of output neuron: 1 / (1 + exp(-0.592605124)) = 0.643962658
Compute an error value by subtracting output from target: 0 - 0.643962658 = **-0.643962658**

Now that we got the output error, let's do the backpropagation.

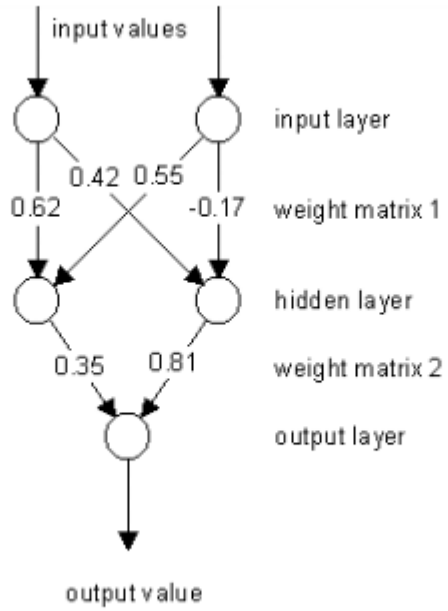

Fig:Backpropagation in 3-layered MultiLayer-Perception

We start with changing the weights in weight matrix 2:

Value for changing weight 1: 0.25 * (-0.643962658) * 0.634135591 * 0.643962658 * (1-0.643962658) =-0.023406638
Value for changing weight 2: 0.25 * (-0.643962658) * 0.457602059 * 0.643962658 * (1-0.643962658) =-0.016890593

Change weight 1: 0.35 + (-0.023406638) = 0.326593362
Change weight 2: 0.81 + (-0.016890593) = 0.793109407

Now we will change the weights in weight matrix 1:

Value for changing weight 1: 0.25 * (-0.643962658) * 0 * 0.634135591 * (1-0.634135591) =0
Value for changing weight 2: 0.25 * (-0.643962658) * 0 * 0.457602059 * (1-0.457602059) =0
Value for changing weight 3: 0.25 * (-0.643962658) * 1 * 0.634135591 * (1-0.634135591) =-0.037351064

Value for changing weight 4: 0.25 * (-0.643962658) * 1 * 0.457602059 * (1-0.457602059) =-0.039958271

Change weight 1: 0.62 + 0 = 0.62 (not changed)
Change weight 2: 0.42 + 0 = 0.42 (not changed)
Change weight 3: 0.55 + (-0.037351064) = 0.512648936
Change weight 4: -0.17+ (-0.039958271) = -0.209958271

The first input pattern had been propagated through the net. The same procedure is used for the next input pattern, but then with the changed weight values.

After the forward and backward propagation of the second pattern, one learning step is complete and the net error can be calculated by adding up the squared output errors of each pattern.

By performing this procedure repeatedly, this error value gets smaller and smaller.

The algorithm is successfully finished, if the net error is zero (perfect) or approximately zero.

FORWARD PROPAGATION :

Forwardpropagation is a supervised learning algorithm and describes the "flow of information" through a neural net from its input layer to its output layer.
The algorithm works as follows:

1. Set all weights to random values ranging from -1.0 to +1.0 .
2. Set an input pattern (binary values) to the neurons of the net's input layer .
3. Activate each neuron of the following layer: - Multiply the weight values of the connections leading to this neuron with the output values of the preceding neurons - Add up these values - Pass the result to an activation function, which computes the output value of this neuron.
4. Repeat this until the output layer is reached.
5. Compare the calculated output pattern to the desired target pattern and compute an error value.

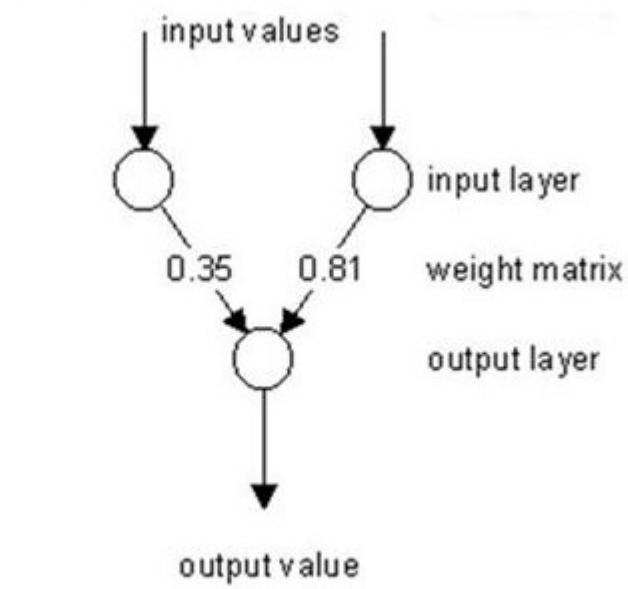

Fig:Forward propagation in a 2-layered Perceptron

6. Change all weights by adding the error value to the (old) weight values.
7. Go to step 2.
8. The algorithm ends, if all output patterns match their target patterns.

Forward Propapagtion : Step-by-Step Example

Suppose you have the 2-layered Perceptron as shown in above figure:
Patterns to be learned:

Input	**Target**
0 $\bar{1}$	0
1 $\bar{1}$	1

First, the weight values are set to random values (0.35 and 0.81).
The learning rate of the net is set to 0.25.
Next, the values of the first input pattern (0 1) are set to the neurons of the input layer (the output of the input layer is the same as its input).

The neurons in the following layer (only one neuron in the output layer) are activated:

Input 1 of output neuron: 0 * 0.35 = 0
Input 2 of output neuron: 1 * 0.81 = 0.81
Add the inputs: 0 + 0.81 = 0.81 (= output)
Compute an error value by
subtracting output from target: 0 - 0.81 = -0.81
Value for changing weight 1: 0.25 * 0 * (-0.81) = 0 (0.25 = learning rate)
Value for changing weight 2: 0.25 * 1 * (-0.81) = -0.2025
Change weight 1: 0.35 + 0 = 0.35 (not changed)
Change weight 2: 0.81 + (-0.2025) = 0.6075 That was one learning step. Each input pattern had been propagated through the net and the weight values were changed.
The error of the net can now be calculated by adding up the squared values of the output errors of each pattern:

Compute the net error: (-0.81)2 + (0.0425)2 = **0.65790625**

By performing this procedure repeatedly, this error value gets smaller and smaller.

The algorithm is successfully finished, if the net error is zero (perfect) or approximately zero.

CONCLUSION :

Thus,we have implemented a simple Neural Network for backward and forward propagation using Java.

Roll No.	**Name of Student**	**Date of Performance**	**Date of Submission**	**Sign.**
		/ /2015	/ /2015	