**Assignment 3**

**Title:** a) To implement XOR gate using backpropogation network (with toolbox) in MATLAB.

b)To implement function interpolator using backpropogation network (without toolbox) in MATLAB.

**Theory:**

Multilayer feed forward networks are an important class of neural networks. Typically, the network consists of sensory units (source nodes) that constitute the input layer, one or more hidden layers of computation nodes and an output layer of computation nodes. The input signal propagates through the network in the forward direction on a layer-by-layer basis. These neural networks are commonly known as Multilayer Perceptron’s (MLPs).

Multilayer perceptron’s have been applied successfully to solve some difficult and diverse problems by training them in a supervised manner with the highly popular algorithm known as the error **back-propagation algorithm**. This algorithm is based on the error-correction learning rule.

Basically error back-propagation learning consists of two passes through the different layers of the network: a forward pass and a backward pass. In the forward pass an activity pattern (input vector) is applied to the sensory nodes of the network and its effect propagates through the network layer by layer. Finally a set of outputs is produced as the actual response of the network. During the forward pass all synaptic weights of the network are fixed. During the backward pass, all synaptic weights are adjusted in accordance with an error correction rule. Specifically the actual response of the network is subtracted from the network desired (target) response to produce an error signal. This error signal is then propagated backward through against the direction of the synaptic weights and hence the name “**error back propagation**”.

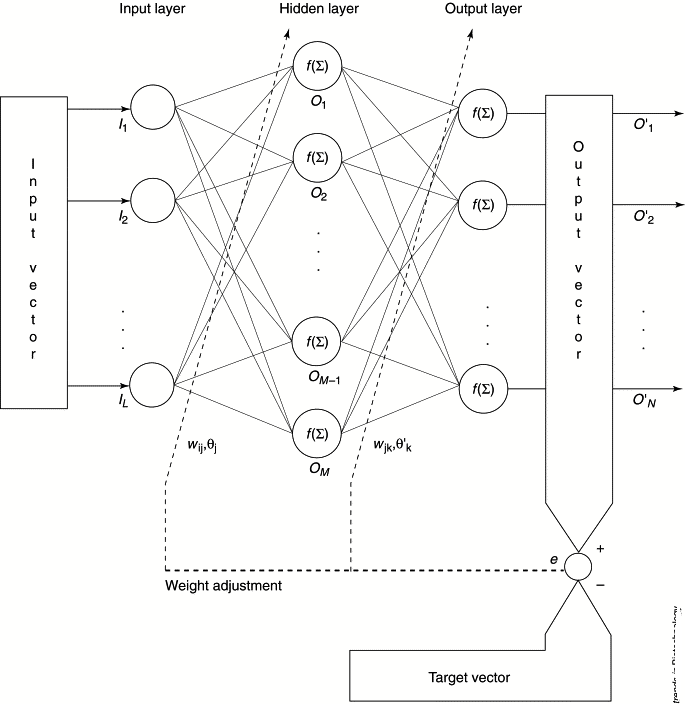
****

Figure 1:Architecture of Backpropogation Network

**Algorithm:**

**Step 0:**  Initialize weights. (Set to small random values)

**Step 1:** While stopping condition is false, do steps 2-9.

**Step 2:** For each training pair, do steps 3-8.

***Feed forward*:**

**Step 3:** Each input unit (Xi, i = 1…….n) receives input signal xi and broadcasts this signal to all units in the hidden layer above (the hidden units.

**Step 4:** Each hidden unit (Zj, j = 1…….p) sums its weighted input signals.

Zinj = voj  + ∑xivij ; i=1…..n

applies its activation function to compute its output signal.

Zj = f(Zinj)

and sends this signal to all units in the layer above (output units).

**Step 5:** Each output unit (Yk =1……m) sums its weighted input signals.

Yink = wok  + ∑zj wjk ; i=1…….n

and applies its activation function to compute its output signal.

***Back propagation of error*:**

**Step 6:** Each output unit receives a target pattern corresponding to the input training

pattern, computes its error information term.

δk = (tk-yk) f’(y\_ink)

calculates its bias correction term(used to update wjk )

∆ wjk  = αδkzj

calculates its bias correction term

∆wok= αδk

and sends δk to units in the layer below.

**Step 7:** Each hidden unit sums its delta inputs (from above in the layer ),

Δ\_inj = ∑k=1m δk wjk,

Multiplies by the derivative of its activation function to calculate its error information term.

Δj= δ\_inj’(z\_inj),

Calculates its weight correction term

∆ vjk  = αδkxj

and calculates its bias correction term

∆voj= αδj

***Update weights and biases***

**Step 8:** Each output unit updates its biases and weights

wjk(new)= wjk(old) + ∆ wjk

each hidden unit updates its biases and weights

vjk(new)= vjk(old) + ∆ vjk

Test stopping condition.

**FAQ’S:**

1. Explain the method to initialize weights for a Backpropogation network.
2. Explain the choice of learning rate parameter.
3. Explain Generalization.
4. How many training data patterns should be used to train a backpropogation network.
5. How to determine the number of Hidden Layer Nodes.