Neural Networks

Lecture 15

Supervised Learning and Backpropagation Neural Networks (3)

Md. Mijanur Rahman, Prof. Dr.

Dept. of Computer Science and Engineering, Jatiya Kabi Kazi Nazrul Islam University, Bangladesh.

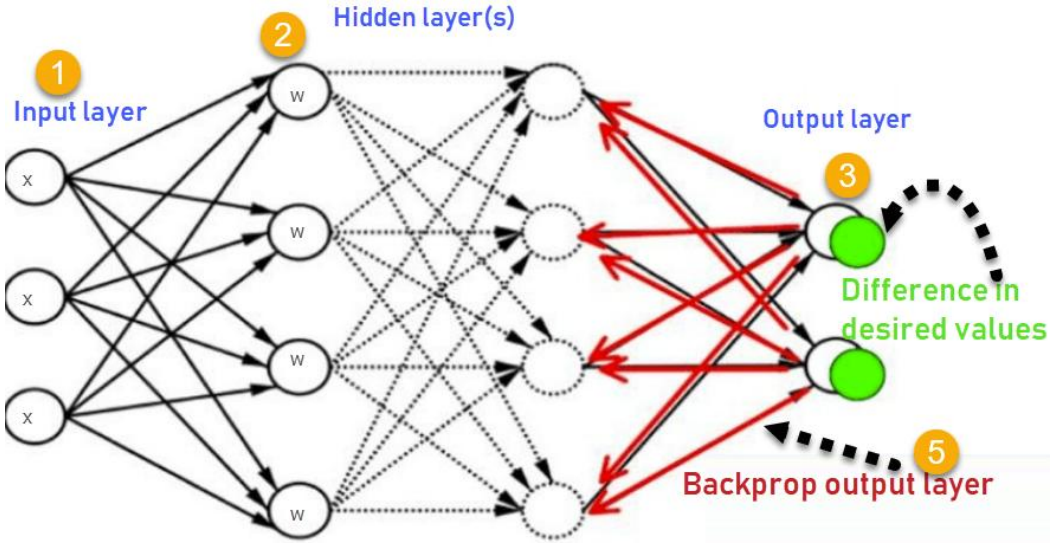
Web: www.mijanrahman.com | Email: mijanjkkniu@gmail.com; mijan@jkkniu.edu.bd

Supervised Learning and Backpropagation Neural Network

- **This chapter covers the following topics:**
  - Supervised Learning Network
  - Perceptron
  - Adaptive Linear Neuron (Adaline)
  - Multiple Adaptive Linear Neuron (Madaline)
  - Generalized Delta Learning Rule
  - Backpropagation Neural Networks
  - Backpropagation Algorithm and Network
  - Types of Activation Functions

Backpropagation Neural Network

- It is a multilayer feedforward neural network, consists of three layers, input, hidden and output layers, that can approximate any smooth, measurable function between input and output vectors by selecting a suitable set of connecting weights and transfer functions.
- The neurons in the same layer are not interlinked while the neurons in adjacent layers are fully connected with weight W and bias Θ.

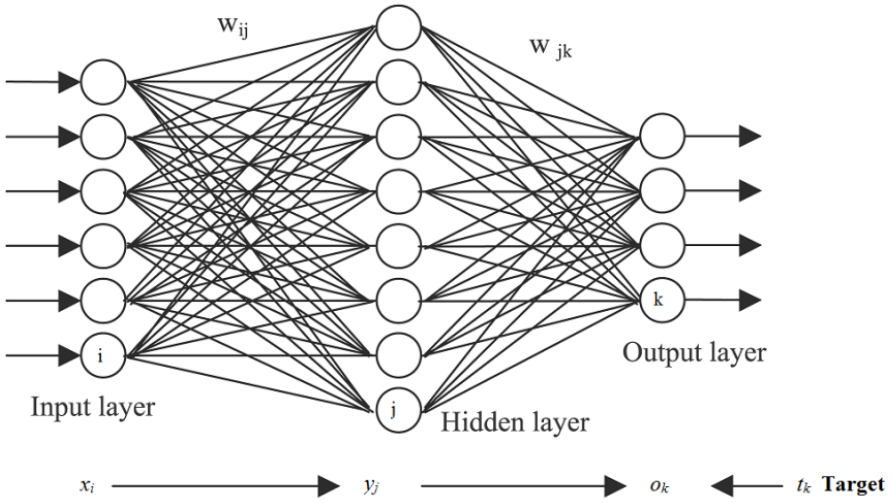


Derivation of BP Algorithm

- The training of a neural network by back-propagation algorithm involves three stages:
  1. The feedforward of the input training pattern (feedforward pass)
  2. The calculation and backpropagation of associated error (backward pass) and
  3. The adjustment of the weights
- The back-propagation algorithm for training multilayer perceptron implements a generalized delta rule.
- The implementation of the generalized delta rule requires to derive an expression for the computation of the derivatives of error with respect to weight

Derivation of BP Algorithm

- ***Feedforward Pass***
  - During Feedforward, each input unit receives an input signal (x_i) and broadcasts this signal to the each hidden units.
  - Each hidden unit computes its activation (y_j) and sends to each output unit.
  - Each output unit computes its activation (o_k) to form the response of the net for the given input pattern.



Feedforward of Input Pattern

Derivation of BP Algorithm

- ***Feedforward Pass***

1. **In the input layer:**
   Each input unit receives an input signal, x_i; $i = 1,\ldots,n$ (where n is number of neurons in input layer) and sends this signal with weights to each hidden unit.
2. **In the hidden layer:**
   Each hidden unit sums its weighted input signal and computes its activation, as follows:

$$y\_in_j = w_{0j} + \sum_{i=1}^{n} w_{ij} x_i; \quad j = 1,\ldots,q$$

$$y_j = f(y\_in_j)$$

where q is the number of neurons in hidden layer, w_{0i} are the biases of the hidden neurons, w_{ij} are the weights between input and hidden layer and f is the activation function.

Derivation of BP Algorithm

- ***Feedforward Pass***

3. In the output layer:

Each output unit sums its weighted hidden output signal and computes its activation, as follows:

$$o\_in_k = w_{0k} + \sum_{j=1}^{q} w_{jk} y_j \; ; \; k = 1, \ldots, p$$

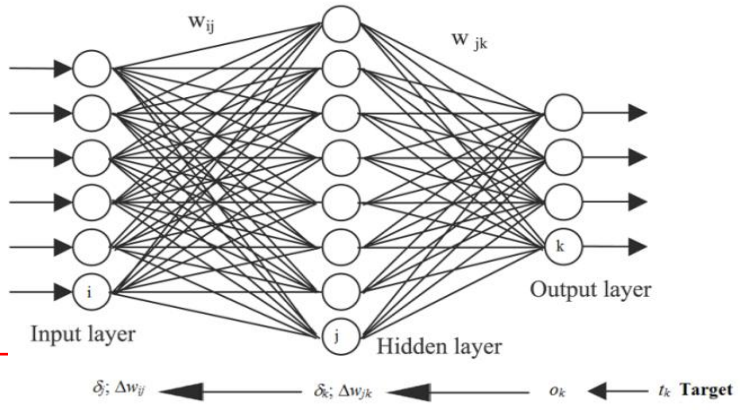
$$o_k = f(o\_in_k)$$

Where p is the number of neurons in output layer, w_{0k} are the biases of the output neurons, w_{jk} are the weights between hidden and output layer and f is the activation function.

Derivation of BP Algorithm

- ***Backward Pass***
  - During calculation and backpropagation of error, each output unit compares its computed activation (o_k) with its target value (t_k) to determine the associated error for that pattern.
  - Based on this error, the factor δ_k is computed. Δ_k is used to distribute the error at output unit (o_k) back to all units in the previous layer (hidden units, connected to output units) and hence the name "error back-propagation".
  - It is also used (later) to update the weights between the output and hidden layer. Similarly, the factor δ_j is computed for each hidden unit (y_j). It is not necessary to propagate the error back to the input layer, but δ_j is used to update the weights between the hidden and input layer.

Error Back-propagation:



Derivation of BP Algorithm

- ***Backward Pass***

1. **Hidden to output layer weight gradients:**

   Each output unit computes its error information term, weight correction term and bias correction term, as follows:

   Error term, $\delta_k = (t_k - o_k) f'(o\_in_k)$

   Weight correction, $\Delta w_{jk} = \eta \delta_k y_j$

   where η is the learning rate.

   Bias correction, $\Delta w_{0k} = \eta \delta_k$

2. **Input to hidden layer weight gradients:**

   Each hidden unit sums its delta inputs, computes its error information term, weight correction term and bias correction term, as follows:

   Summation of delta inputs, $\delta\_in_j = \sum_{k=1}^{p} \delta_k w_{jk}$

   Error term, $\delta_j = \delta\_in_j f'(y\_in_j)$

   Weight correction, $\Delta w_{ij} = \eta \delta_j x_i$

   Bias correction, $\Delta w_{0j} = \eta \delta_j$

Derivation of BP Algorithm

- ***Adjustment of Weights***
  - After all of the δ factors have been determined, the weights of all layers are adjusted simultaneously.

A. The adjustment of weights w_{jk} (from hidden to output unit) is based on the factor δ_k and the activation of hidden unit (y_j).

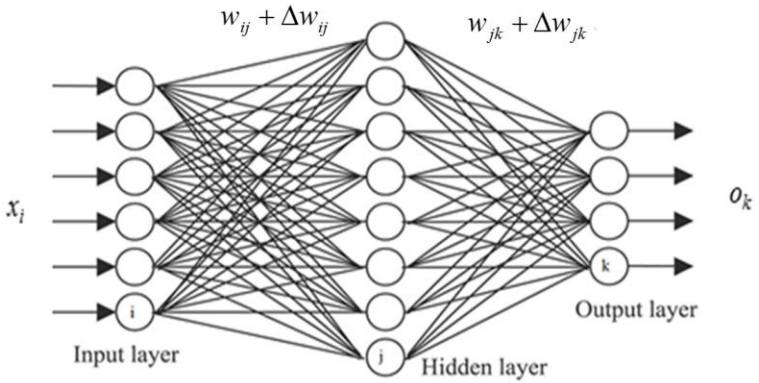
B. The adjustment of weights w_{ij} (from input to hidden unit) is based on the factor δ_j and the activation of the input unit (x_i).

1. For hidden to output layer weights:

$$w_{jk}(new) = w_{jk} + \Delta w_{jk} = w_{jk} + \eta \delta_k y_j$$

2. For input to hidden layer weights:

$$w_{ij}(new) = w_{ij} + \Delta w_{ij} = w_{ij} + \eta \delta_j x_i$$



This is the complete derivation of the back-propagation algorithm.

Overall BP Algorithm

- **The basic procedure of gradient descent based training is outlines as follows in the steps 1-6.**

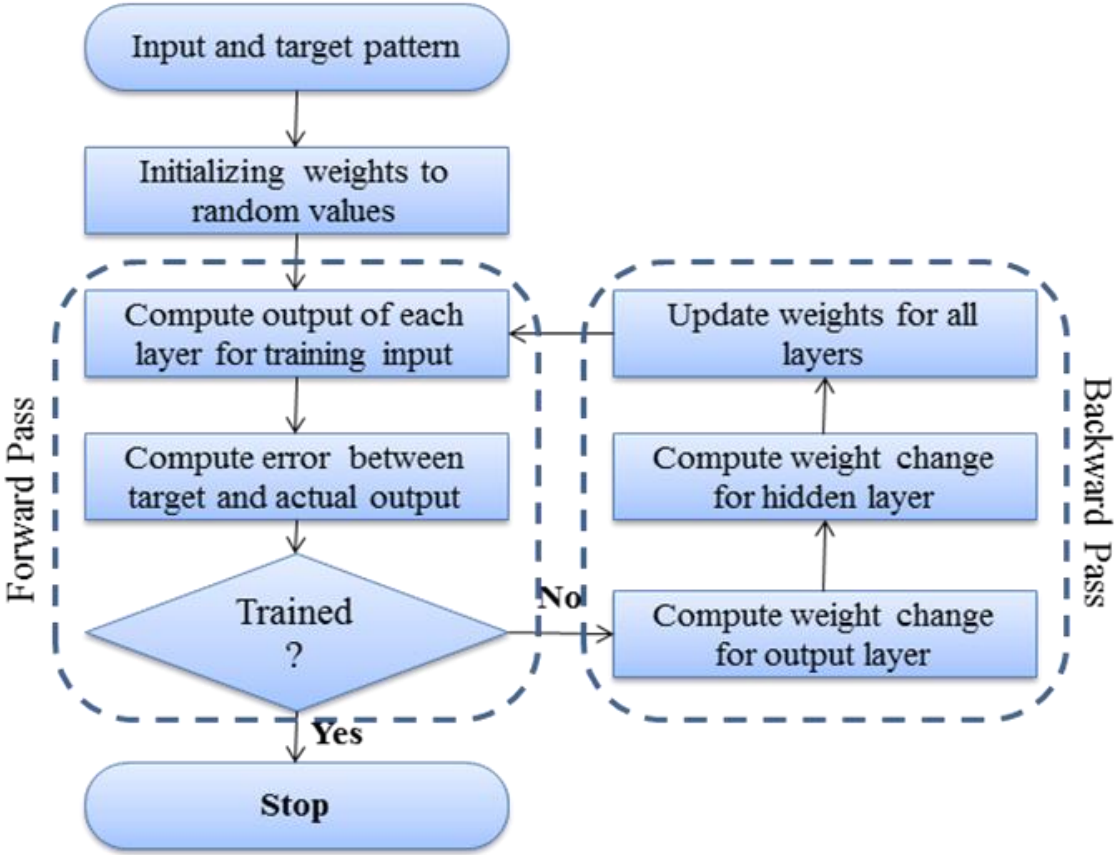
1. Select input pattern, x_i, initialize weights (set to small random values), and present it to the network.
2. Sum weighted signals and compute its activation of input, hidden and output neurons in that sequence.
3. Compute the error over the output neurons by comparing the generated outputs with the target outputs.
4. Use the error calculated from step 3 to compute the change in the hidden to output layer weights, and the change in input to hidden layer weights, such that a global error measure gets reduced.
5. Update all weights of the network in accordance with the changes computed in step 4.

   Hidden to output layer weights: $w_{jk}(new) = w_{jk} + \Delta w_{jk}$

   Input to hidden layer weights: $w_{ij}(new) = w_{ij} + \Delta w_{ij}$
6. Repeat steps 2 to 5 until a stopping criterion hold (e.g. the global error falls below a predefined threshold).

Overall BP Algorithm

- **Back-propagation Training Algorithm:**

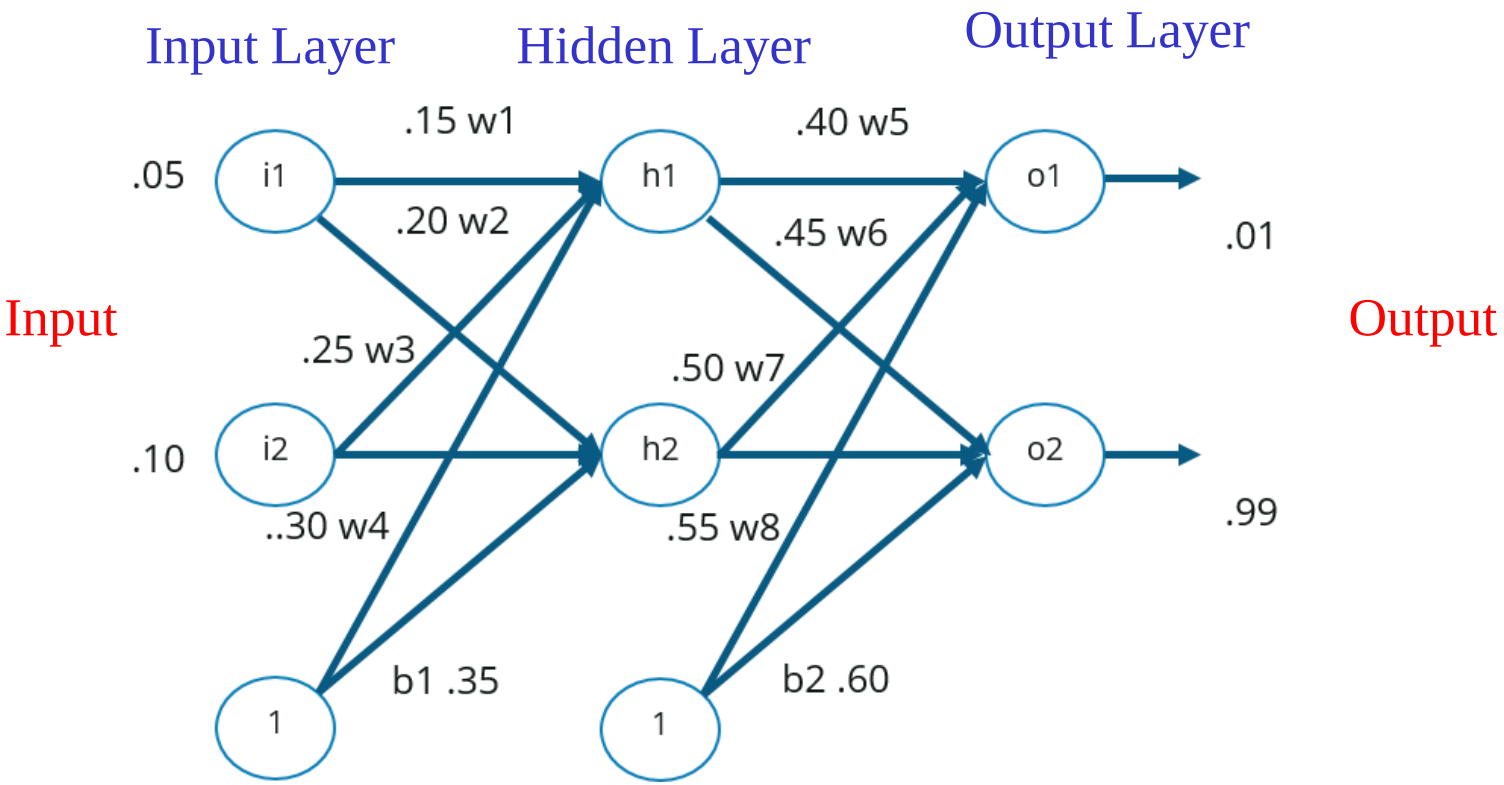


Overall BP Algorithm

- **Back-propagation Training Algorithm: Stopping Criteria**
  - The stopping criteria of the training algorithm are checked at the end of each epoch:
    - The error (or mean square) at the end of an epoch is below a threshold.
      - The threshold is determined heuristically (For example, 0.3).
    - Maximum number of epochs is reached.
  - It typically takes hundreds or thousands of epochs for a neural network to converge.

How BP Algorithm Works?

- Consider the below Neural Network with 2 neurons in input layer, 2 neurons in hidden layer, and 2 output neurons:



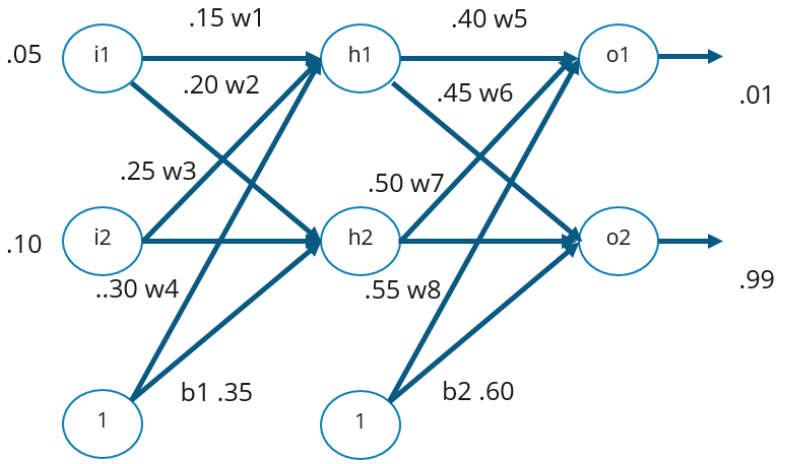
How BP Algorithm Works?

- Initially, the network contains the following parameters:
  - **Inputs**, X={0.05, 0.10}; **Weights**, W_{ih}={0.15,0.20; 0.25, 0.30}, W_{ho}={0.40, 0.45; 0.50, 0.55}; **Biases**, B={0.35, 0.60}
- Below are the steps involved in Backpropagation algorithm:

Step – 1: Forward Propagation

Step – 2: Backward Propagation

Step – 3: Putting all the values together and calculating the updated weight value



How BP Algorithm Works?

Step – 1: Forward Propagation

- It will be started by propagating forward.

Net Input For h1:

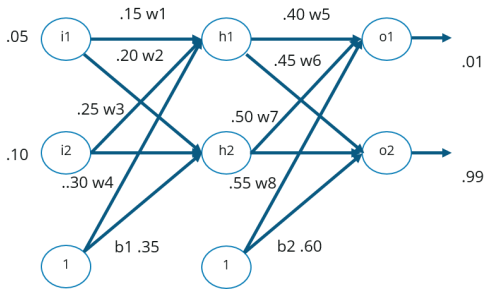
net h1 = w1*i1 + w2*i2 + b1*1 → net h1 = 0.15*0.05 + 0.2*0.1 + 0.35*1 = 0.3775

Output Of h1:

out h1 = $1/1+e^{-\text{net h1}}$ → $1/1+e^{.3775}$ = 0.593269992

Output Of h2:

out h2 = 0.596884378



How BP Algorithm Works?

Step – 1: Forward Propagation

- Repeat this process for the output layer neurons, using the output from the hidden layer neurons as inputs.

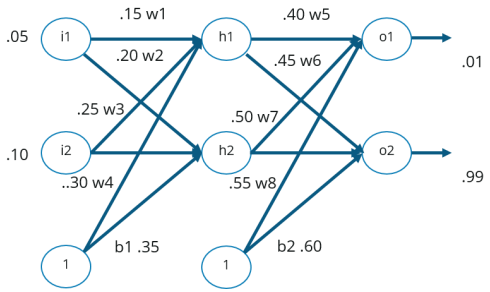
Output For o1:

net o1 = w5*out h1 + w6*out h2 + b2*1 → 0.4*0.593269992 + 0.45*0.596884378 + 0.6*1 = 1.105905967

Out o1 = $1/1+e^{-\text{net o1}}$ → $1/1+e^{-1.105905967}$ = 0.75136507

Output For o2:

Out o2 = 0.772928465



How BP Algorithm Works?

Step – 1: Forward Propagation

- Compute the value of the error:

Error For o1:

$E_{o1} = \Sigma 1/2(target - output)^2$ → $\frac{1}{2}(0.01 - 0.75136507)^2 = 0.274811083$

Error For o2:

$E_{o2} = 0.023560026$

Total Error:

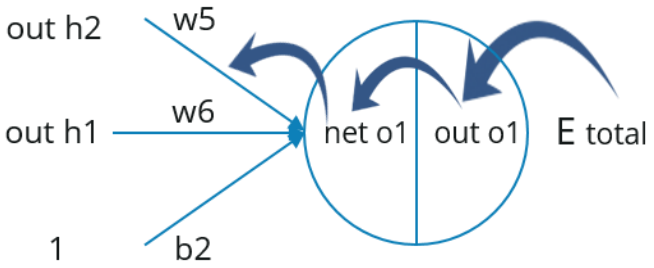
$E_{total} = E_{o1} + E_{o2}$ → $0.274811083 + 0.023560026 = 0.298371109$

How BP Algorithm Works?

Step – 2: Backward Propagation

- It will be propagated backwards. This way we will try to reduce the error by changing the values of weights and biases.
- Consider W5, we will calculate the rate of change of error w.r.t change in weight W5.

$$\frac{\delta Etotal}{\delta w5} = \frac{\delta Etotal}{\delta out\ o1} * \frac{\delta out\ o1}{\delta net\ o1} * \frac{\delta net\ o1}{\delta w5}$$



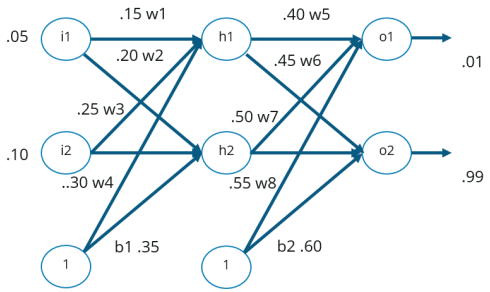
How BP Algorithm Works?

Step – 2: Backward Propagation

- Since we are propagating backwards, first thing we need to do is, calculate the change in total errors w.r.t the output O1 and O2.

$$E_{total} = 1/2(\text{target o1} - \text{out o1})^2 + 1/2(\text{target o2} - \text{out o2})^2$$

$$\frac{\delta Etotal}{\delta out\ o1} = -(\text{target o1} - \text{out o1}) = -(0.01 - 0.75136507) = 0.74136507$$



How BP Algorithm Works?

Step – 2: Backward Propagation

- Now, we will propagate further backwards and calculate the change in output O1 w.r.t to its total net input.

$$\text{out o1} = 1/1+e^{-neto1}$$

$$\frac{\delta out\ o1}{\delta net\ o1} = \text{out o1 (1 - out o1)} = 0.75136507\ (1 - 0.75136507) = 0.186815602$$

- Let's see now how much does the total net input of O1 changes w.r.t W5?

$$\text{net o1} = w5 * \text{out h1} + w6 * \text{out h2} + b2 * 1$$

$$\frac{\delta net\ o1}{\delta w5} = 1 * \text{out h1}\ w5^{(1-1)} + 0 + 0 = 0.593269992$$

How BP Algorithm Works?

Step – 3: Putting all the values together and calculating the updated weight value

- Now, let's put all the values together:

$$\frac{\delta Etotal}{\delta w5} = \frac{\delta Etotal}{\delta out\ o1} * \frac{\delta out\ o1}{\delta net\ o1} * \frac{\delta net\ o1}{\delta w5}$$

→ 0.082167041

- Let's calculate the updated value of W5:

$$w5^{+} = w5 - n\frac{\delta Etotal}{\delta w5}$$

→ $w5^{+} = 0.4 - 0.5 * 0.082167041$

Updated w5 → 0.35891648

How BP Algorithm Works?

- Similarly, we can calculate the other weight values as well.
- After that we will again propagate forward and calculate the output. Again, we will calculate the error.
- If the error is minimum we will stop right there, else we will again propagate backwards and update the weight values.
- This process will keep on repeating until error becomes minimum.

SUPERVISED LEARNING AND BACKPROPAGATION NETWORK PARADIGMS

To be continued…