D.P.VIPRA COLLEGE BILASPUR (C.G)

Topic: Backpropagation Algorithm

Presented by:
Miss. Neelam Singh Singhel

Backpropagation:

Backpropagation is a widely used algorithm for training feedforward neural networks. It computes the gradient of the loss function with respect to the network weights and is very efficient, rather than naively directly computing the gradient with respect to each individual weight. This efficiency makes it possible to use gradient methods to train multi-layer networks and update weights to minimize loss; variants such as gradient descent or stochastic gradient descent are often used.

The backpropagation algorithm works by computing the gradient of the loss function with respect to each weight via the chain rule, computing the gradient layer by layer, and iterating backward from the last layer to avoid redundant computation of intermediate terms in the chain rule.

Working of Backpropagation:
Neural networks use supervised learning to generate output vectors from input vectors that the network operates on. It Compares generated output to the desired output and generates an error report if the result does not match the generated output vector. Then it adjusts the weights according to the bug report to get your desired output.

Backpropagation Algorithm:

Step 1: Inputs X, arrive through the preconnected path.
Step 2: The input is modeled using true weights W. Weights are usually chosen randomly.
Step 3: Calculate the output of each neuron from the input layer to the hidden layer to the output layer.
Step 4: Calculate the error in the outputs
Backpropagation Error= Actual Output – Desired Output

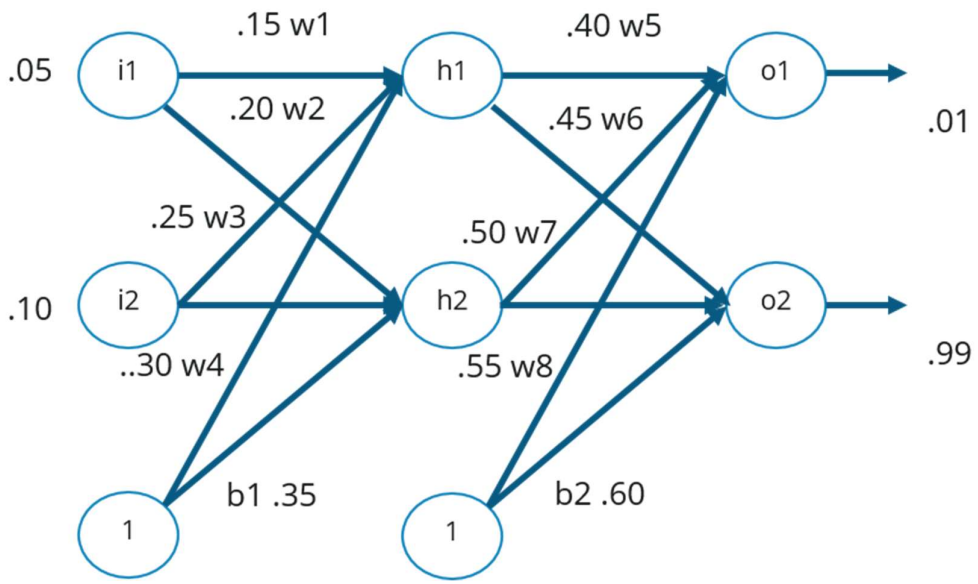
Step 5: From the output layer, go back to the hidden layer to adjust the weights to reduce the error.
Step 6: Repeat the process until the desired output is achieved.

Need for Backpropagation:

Backpropagation is “backpropagation of errors” and is very useful for training neural networks. It’s fast, easy to implement, and simple. Backpropagation does not require any parameters to be set, except the number of inputs. Backpropagation is a flexible method because no prior knowledge of the network is required.

How Backpropagation Works?

Consider the below Neural Network:



The above network contains the following:

- two inputs
- two hidden neurons
- two output neurons
- two biases

Below are the steps involved in Backpropagation:

- Step – 1: Forward Propagation
- Step – 2: Backward Propagation
- Step – 3: Putting all the values together and calculating the updated weight value

Step – 1: Forward Propagation

We will start 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

We will 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

Now, let's see what is the value of the error:

Error For o1:

E o1 = $\Sigma 1/2(target - output)^2$ → ½ (0.01 – 0.75136507)² = 0.274811083

Error For o2:

E o2 = 0.023560026

Total Error:

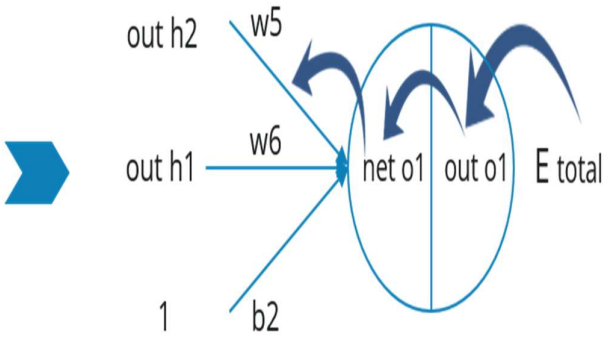
E_{total} = E o1 + E o2 → 0.274811083 + 0.023560026 = 0.298371109

Step – 2: Backward Propagation

Now, we will propagate 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.

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



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.

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

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

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

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

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

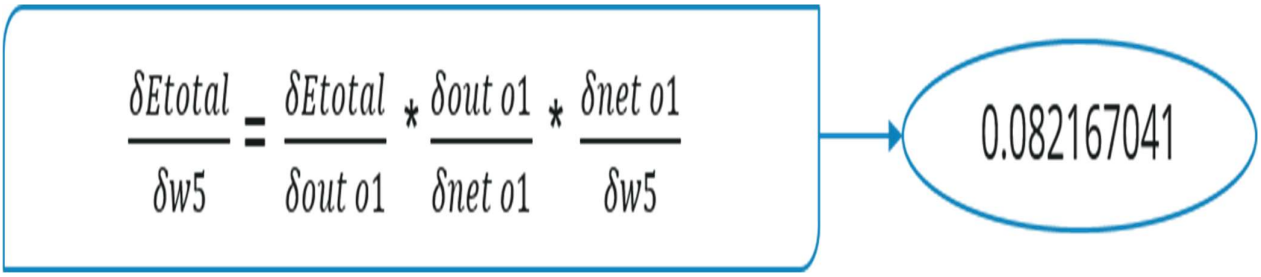
Let's see now how much does the total net input of O1 changes w.r.t W5?

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

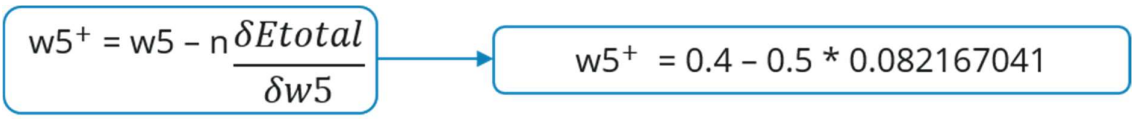
$$\frac{\delta net\ o1}{\delta w5} = 1 * out\ h1\ w5^{(1-1)} + 0 + 0 = 0.593269992$$

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

Now, let's put all the values together:



Let's calculate the updated value of W5:



- 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.

Conclusion:
Well, if I have to conclude Backpropagation, the best option is to write pseudo code for the same.

Backpropagation Algorithm:
initialize network weights (often small random values)