How does backpropagation work?

Backpropagation defines the whole procedure encompassing both the computation of the gradient and its need in the stochastic gradient descent. Technically, backpropagation is used to calculate the gradient of the error of the network with respect to the network's modifiable weights.

The characteristics of Backpropagation are the iterative, recursive and effective approach through which it computes the updated weight to enhance the network until it cannot perform the function for which it is being trained. Derivatives of the activation service to be known at web design time is needed to Backpropagation.

Backpropagation is generally used in neural network training and computes the loss function concerning the weights of the network. It functions with a multi-layer neural network and observes the internal representations of input-output mapping.

It is a standard form of artificial network training, which helps to calculate gradient loss function with respect to all weights in the network. The backpropagation algorithm is used to train a neural network more effectively through a chain rule method. It defines after each forward, the backpropagation implements backward pass through a web by adjusting the arguments of the model.

This gradient is used in simple stochastic gradient descent algorithm to find weights that minimizes the error. The error propagate backwards from the output nodes to the inner nodes.

Backpropagation understand by iteratively processing a data collection of training tuples, comparing the network's indicator for every tuple with the actual known target value. The target value can be the known class label of the training tuple (for classification issues) or a continuous value (for prediction).

For each training tuple, the weights are modified so as to minimize the mean squared error between the network's prediction and the actual target value. These modifications are made in the "backwards" direction, that is, from the output layer, through each hidden layer down to the first hidden layer (hence the name backpropagation). Although it is not protected, in general the weights will finally assemble, and the learning process end.

Types of Backpropagation

There are two types of Back propagation which are as follows −

Static Back Propagation − In this type of backpropagation, the static output is generated due to the mapping of static input. It can resolve static classification issues like optical character recognition.

Recurrent Backpropagation − The Recurrent Propagation is directed forward or conducted until a certain determined value or threshold value is reached. After the specific value, the bug is computed and propagated backward.

Why We Need Backpropagation?

While designing a Neural Network, in the beginning, we initialize weights with some random values or any variable for that fact.

Now obviously, we are not *superhuman.* So, it's not necessary that whatever weight values we have selected will be correct, or it fits our model the best.

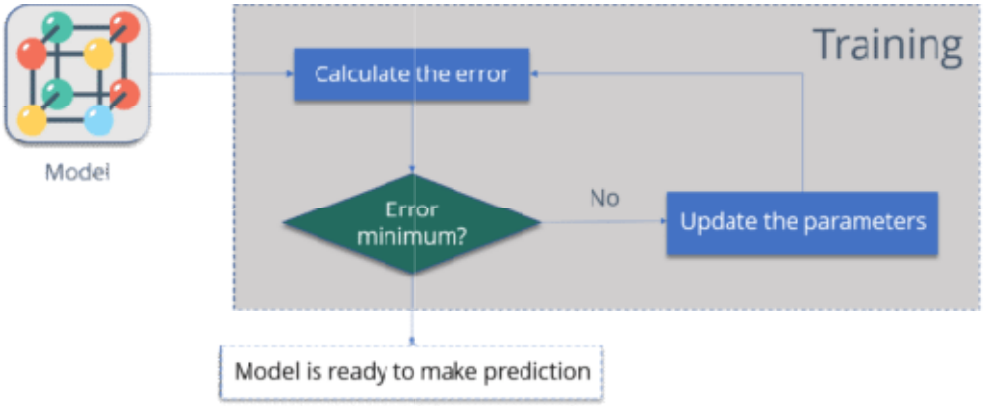
Okay, fine, we have selected some weight values in the beginning, but our model output is way different than our actual output i.e. the error value is huge.

Now, how will you reduce the error?

Basically, what we need to do, we need to somehow explain the model to change the parameters (weights), such that error becomes minimum.

Let's put it in an another way, we need to train our model.

One way to train our model is called as Backpropagation. Consider the diagram below:



Let summarize the steps:

- **Calculate the error** – How far is your model output from the actual output.
- **Minimum Error** – Check whether the error is minimized or not.
- **Update the parameters** – If the error is huge then, update the parameters (weights and biases). After that again check the error. Repeat the process until the error becomes minimum.
- **Model is ready to make a prediction** – Once the error becomes minimum, you can feed some inputs to your model and it will produce the output.

I am pretty sure, now you know, why we need Backpropagation or why and what is the meaning of training a model.

Now is the correct time to understand what is Backpropagation.

What is Backpropagation?

The Backpropagation algorithm looks for the minimum value of the error function in weight space using a technique called the delta rule or gradient descent. The weights that minimize the error function is then considered to be a solution to the learning problem.

Let's understand how it works with an example:

You have a dataset, which has labels.

Consider the below table:

Input	Desired Output
0	0
1	2

2	4

Now the output of your model when ‘W” value is 3:

Input	Desired Output	Model output (W=3)
0	0	0
1	2	3
2	4	6

Notice the difference between the actual output and the desired output:

Input	Desired Output	Model output (W=3)	Absolute Error	Square Error
0	0	0	0	0
1	2	3	1	1
2	4	6	2	4

Let’s change the value of ‘W’. Notice the error when ‘W’ = ‘4’

Input	Desired Output	Model output (W=3)	Absolute Error	Square Error	Model output (W=4)	Square Error
0	0	0	0	0	0	0
1	2	3	1	1	4	4
2	4	6	2	4	8	16

Now if you notice, when we increase the value of ‘W’ the error has increased. So, obviously there is no point in increasing the value of ‘W’ further. But, what happens if I decrease the value of ‘W’? Consider the table below:

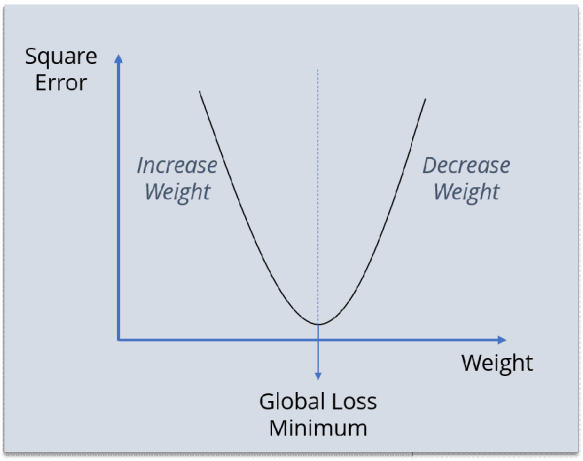
Input	Desired Output	Model output (W=3)	Absolute Error	Square Error	Model output (W=2)	Square Error
0	0	0	0	0	0	0
1	2	3	2	4	3	0
2	4	6	2	4	4	0

Now, what we did here:

- We first initialized some random value to ‘W’ and propagated forward.
- Then, we noticed that there is some error. To reduce that error, we propagated backwards and increased the value of ‘W’.
- After that, also we noticed that the error has increased. We came to know that, we can’t increase the ‘W’ value.
- So, we again propagated backwards and we decreased ‘W’ value.
- Now, we noticed that the error has reduced.

So, we are trying to get the value of weight such that the error becomes minimum. Basically, we need to figure out whether we need to increase or decrease the weight value. Once we know that, we keep on updating the weight value in that direction until error becomes minimum. You might reach a point, where if you further update the weight, the error will increase. At that time you need to stop, and that is your final weight value.

Consider the graph below:



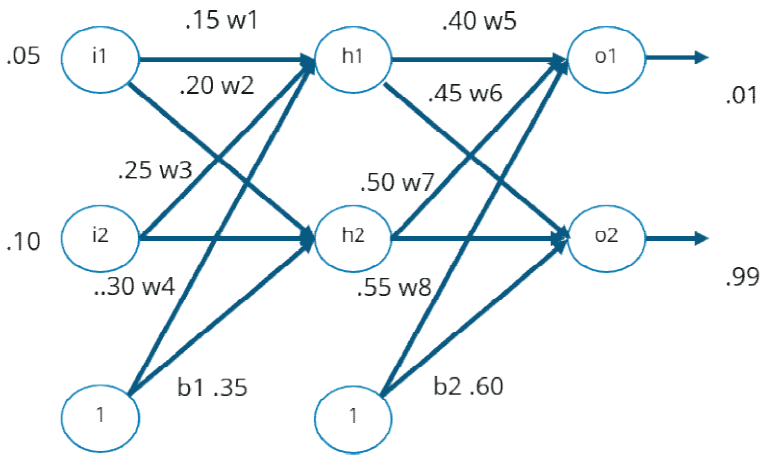
We need to reach the ‘Global Loss Minimum’.

This is nothing but Backpropagation.

Let’s now understand the math behind Backpropagation.

How Backpropagation Works?

Consider the below Neural Network:



The above network contains the following:

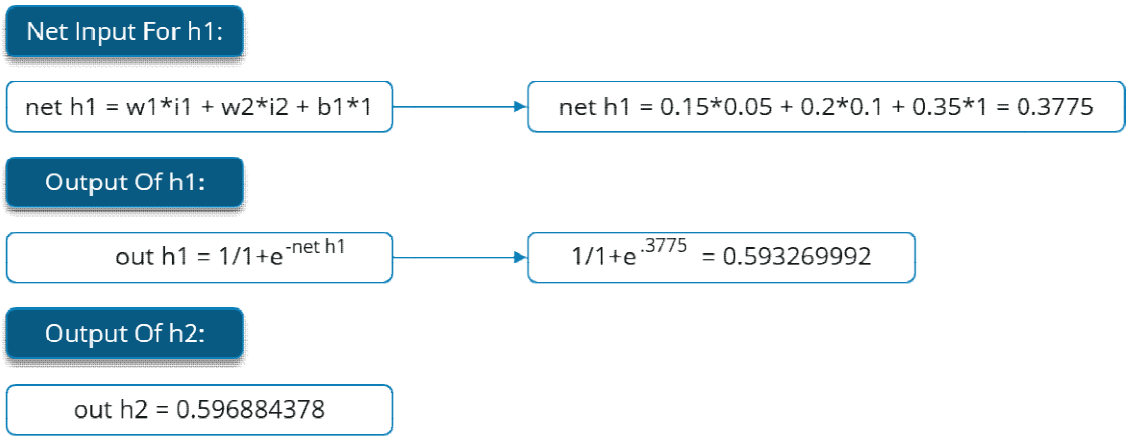
- two inputs
- two hidden neurons
- two output neurons
- two biases

Below are the steps involved in Backpropagation:

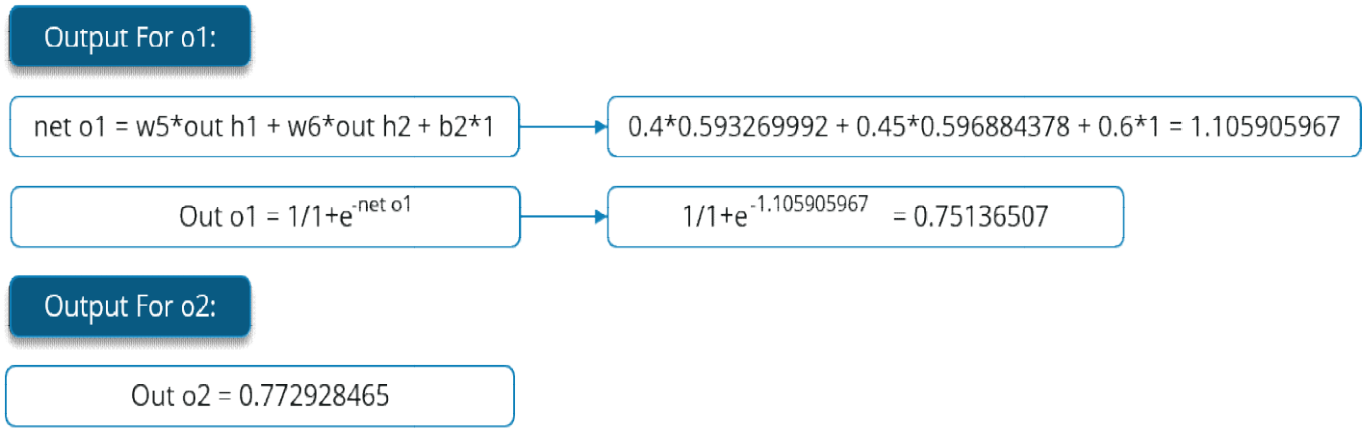
- Step – 1: Forward Propagation
- Step – 2: Backward Propagation
- Step – 3: Putting all the values together and calculating the updated weight value

Step – 1: Forward Propagation

We will start by propagating forward.



We will repeat this process for the output layer neurons, using the output from the hidden layer neurons as inputs.



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:

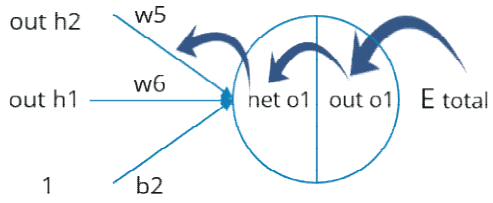
E_{total} = E o1 + E o2 → 0.274811083 + 0.023560026 = 0.298371109

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.

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

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

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:

$$\frac{\delta Etotal}{\delta w5} = \frac{\delta Etotal}{\delta out\ o1} * \frac{\delta out\ o1}{\delta net\ o1} * \frac{\delta net\ o1}{\delta w5} \rightarrow 0.082167041$$

Let's calculate the updated value of W5:

$$w5^{+} = w5 - n\frac{\delta Etotal}{\delta w5} \rightarrow w5^{+} = 0.4 - 0.5 * 0.082167041$$

Updated w5 → 0.35891648

- 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)

do

forEach training example named ex

prediction = neural-net-output(network, ex) *// forward pass*

actual = teacher-output(ex)

compute error (prediction - actual) at the output units

compute {displaystyle Delta w_{h}} for all weights from hidden layer to output layer *// backward pass*

compute {displaystyle Delta w_{i}} for all weights from input layer to hidden layer *// backward pass continued*

update network weights *// input layer not modified by error estimate*

until all examples classified correctly or another stopping criterion satisfied

return the network