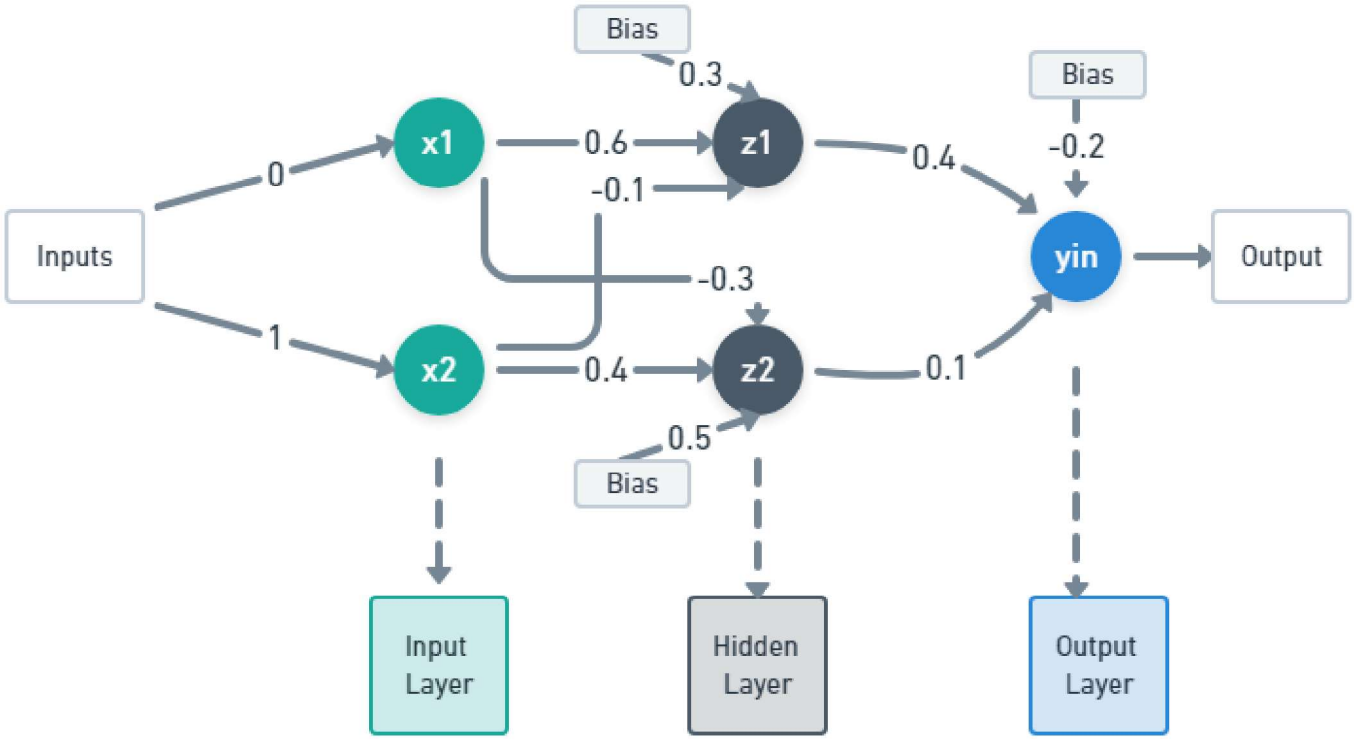
Understanding Multi-Layer Feed Forward Networks

Let's understand how errors are calculated and weights are updated in backpropagation networks(BPNs).

Consider the following network in the below figure.



Backpropagation Network (BPN)

The network in the above figure is a simple multi-layer feed-forward network or backpropagation network. It contains three layers, the input layer with two neurons x1 and x2, the hidden layer with two neurons z1 and z2 and the output layer with one neuron yin.

Now let's write down the weights and bias vectors for each neuron.

Note: The weights are taken randomly.

Input layer: i/p – [x1 x2] = [0 1]

Here since it is the input layer only the input values are present.

Hidden layer: z1 – [v11 v21 v01] = [0.6 -0.1 03]

Here v11 refers to the weight of first input x1 on z1, v21 refers to the weight of second input x2 on z1 and v01 refers to the bias value on z1.

z2 **–** [v12 v22 v02] = [-0.3 0.4 0.5]

Here v12 refers to the weight of first input x1 on z2, v22 refers to the weight of second input x2 on z2 and v02 refers to the bias value on z2.

Output layer: yin **–** [w11 w21 w01] = [0.4 0.1 -0.2]

Here w11 refers to the weight of first neuron z1 in a hidden layer on yin, w21 refers to the weight of second neuron z2 in a hidden layer on yin and w01 refers to the bias value on yin. Let's consider three variables, k which refers to the neurons in the output layer, 'j' which refers to the neurons in the hidden layer and 'i' which refers to the neurons in the input layer.

Therefore,

k = 1

j = 1, 2(meaning first neuron and second neuron in hidden layer)

i = 1, 2(meaning first and second neuron in the input layer)

Below are some conditions to be followed in BPNs.

Conditions/Constraints:

1. In BPN, the activation function used should be differentiable.
2. The input for bias is always 1.

To proceed with the problem, let:

Target value, t = 1

Learning rate, α = 0.25

Activation function = Binary sigmoid function

Binary sigmoid function, f(x) = (1+e-x)-1 eq. (1)

And, f'(x) = f(x)[1-f(x)] eq. (2)

There are three steps to solve the problem:

1. Computing the output, y.
2. Backpropagation of errors, i.e., between output and hidden layer, hidden and input layer.

3. Updating weights.

Step 1:

The value y is calculated by finding yin and applying the activation function.

yin is calculated as:

yin = w01 + z1*w11 + z2*w21 eq. (3)

Here, z1 and z2 are the values from hidden layer, calculated by finding zin1, zin2 and applying activation function to them.

zin1 and zin2 are calculated as:

zin1 = v01 + x1*v11 + x2*v21 eq. (4)

zin2 = v02 + x1*v12 + x2*v22 eq. (5)

From (4)

zin1 = 0.3 + 0*0.6 + 1*(-0.1)

zin1 = 0.2

z1 = f(zin1) = (1+e-0.2)-1 From (1)

z1 = 0.5498

From (5)

zin2 = 0.5 + 0*(-0.3) + 1*0.4

zin2 = 0.9

z2 = f(zin2) = (1+e-0.9)-1 From (1)

z2 = 0.7109

From (3)

yin = (-0.2) + 0.5498*0.4 + 0.7109*0.1

yin = 0.0910

y = f(yin) = (1+e-0.0910)-1 From (1)

y = 0.5227

Here, y is not equal to the target 't', which is 1. And we proceed to calculate the errors and then update weights from them in order to achieve the target value.

Step 2:

(a) Calculating the error between output and hidden layer

Error between output and hidden layer is represented as δk, where k represents the neurons in output layer as mentioned above. The error is calculated as:

$\delta k = (tk - yk) * f'(yink)$ eq. (6)

where, $f'(yink) = f(yink)[1 - f(yink)]$ From (2)

Since k = 1 (Assumed above),

$\delta = (t - y) f'(yin)$ eq. (7)

where, $f'(yin) = f(yin)[1 - f(yin)]$

$f'(yin) = 0.5227[1 - 0.5227]$

$f'(yin) = 0.2495$

Therefore,

$\delta = (1 - 0.5227) * 0.2495$ From (7)

δ = 0.1191, is the error

Note: (Target – Output) i.e., (t – y) is the error in the output not in the layer. Error in a layer is contributed by different factors like weights and bias.

(b) Calculating the error between hidden and input layer

Error between hidden and input layer is represented as δj, where j represents the number of neurons in the hidden layer as mentioned above. The error is calculated as:

$\delta j = \delta inj * f'(zinj)$ eq. (8)

where,

$\delta inj = \sum k=1 \text{ to } n (\delta k * wjk)$ eq. (9)

$f'(zinj) = f(zinj)[1 - f(zinj)]$ eq. (10)

Since k = 1(Assumed above) eq. (9) becomes:

$\delta inj = \delta * wj1$ eq. (11)

As j = 1, 2, we will have one error values for each neuron and total of 2 errors values.

$\delta 1 = \delta in1 * f'(zin1)$ eq. (12), From (8)

$\delta in1 = \delta * w11$ From (11)

$\delta in1 = 0.1191 * 0.4$ From weights vectors

$\delta in1 = 0.04764$

$f'(zin1) = f(zin1)[1 - f(zin1)]$

$f'(zin1) = 0.5498[1 - 0.5498]$ As f(zin1) = z1

$f'(zin1) = 0.2475$

Substituting in (12)

$\delta 1 = 0.04674 * 0.2475 = 0.0118$

$\delta 2 = \delta in2 * f'(zin2)$ eq. (13), From (8)

$\delta in2 = \delta * w21$ From (11)

$\delta in2 = 0.1191 * 0.1$ From weights vectors

$\delta in2 = 0.0119$

$f'(zin2) = f(zin2)[1 - f(zin2)]$

$f'(zin2) = 0.7109[1 - 0.7109]$ As f(zin2) = z2

$f'(zin2) = 0.2055$

Substituting in (13)

$\delta 2 = 0.0119 * 0.2055 = 0.00245$

The errors have been calculated, the weights have to be updated using these error values.

Step 3:

The formula for updating weights for output layer is:

$wjk(new) = wjk(old) + \Delta wjk$ eq. (14)

where, $\Delta wjk = \alpha * \delta k * zj$ eq. (15)

Since k = 1, (15) becomes:

$\Delta wjk = \alpha * \delta * zi$ eq. (16)

The formula for updating weights for hidden layer is:

$vij(new) = vij(old) + \Delta vij$ eq. (17)

where, $\Delta vi = \alpha * \delta j * xi$ eq. (18)

From (14) and (16)

$w11(new) = w11(old) + \Delta w11 = 0.4 + \alpha * \delta * z1 = 0.4 + 0.25 * 0.1191 * 0.5498 = 0.4164$

$w21(new) = w21(old) + \Delta w21 = 0.1 + \alpha * \delta * z2 = 0.1 + 0.25 * 0.1191 * 0.7109 = 0.12117$

$w01(new) = w01(old) + \Delta w01 = (-0.2) + \alpha * \delta * bias = (-0.2) + 0.25 * 0.1191 * 1 = -0.1709$, kindly note the 1 taken here is input considered for bias as per the conditions.

These are the updated weights of the output layer.

From (17) and (18)

$v11(new) = v11(old) + \Delta v11 = 0.6 + \alpha * \delta 1 * x1 = 0.6 + 0.25 * 0.0118 * 0 = 0.6$

$v21(new) = v21(old) + \Delta v21 = (-0.1) + \alpha * \delta 1 * x2 = (-0.1) + 0.25 * 0.0118 * 1 = 0.00295$

$v01(new) = v01(old) + \Delta v01 = 0.3 + \alpha * \delta 1 * bias = 0.3 + 0.25 * 0.0118 * 1 = 0.00295$, kindly note the 1 taken here is input considered for bias as per the conditions.

$v12(new) = v12(old) + \Delta v12 = (-0.3) + \alpha * \delta 2 * x1 = (-0.3) + 0.25 * 0.00245 * 0 = -0.3$

$v22(new) = v22(old) + \Delta v22 = 0.4 + \alpha * \delta 2 * x2 = 0.4 + 0.25 * 0.00245 * 1 = 0.400612$

$v02(new) = v02(old) + \Delta v02 = 0.5 + \alpha * \delta 2 * bias = 0.5 + 0.25 * 0.00245 * 1 = 0.500612$, kindly note the 1 taken here is input considered for bias as per the conditions.

These are all the updated weights of the hidden layer.

These three steps are repeated until the output ‘y’ is equal to the target ‘t’.

This is how the BPNs work. The backpropagation in BPN refers to that the error in the present layer is used to update weights between the present and previous layer by backpropagating the error values.