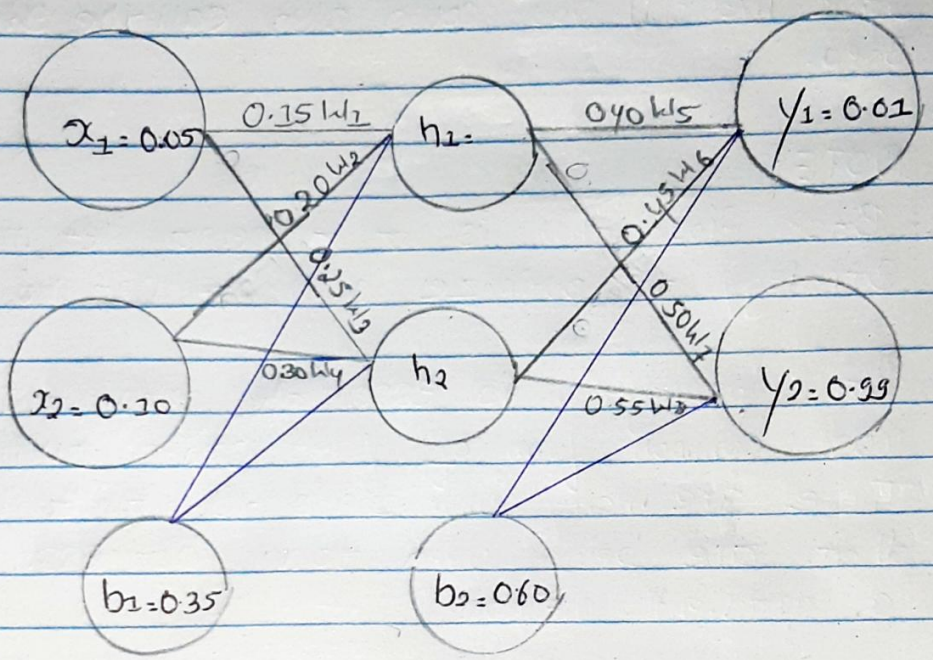
AI and Machine Learning

** A Step by Step Backpropagation **



1. What is back propagation in neural networks?

→ When you use a neural network, the inputs are processed by the (ahem) neurons using certain weights to yield the output. This is like a signal propagating through the network. When training the network, you generate an error signal when the inputs are 'propagated' through to the outputs (usually the difference between outputs and the expected known values). Now the errors are used to change the weights, that is, the errors are processed to generate a change in the weights, also called an update. It's like the errors are propagating backwards through the network to yields a better set of weights that would match the inputs to the output, at least on the training data. That's a crude way to understand

back propagation. The back propagation step yields new weights for the neurons, through the process of optimization via gradient descent (this is the most basic method). You could also call the step a feedback step.

NOTE:

① Back Propagation is not a neural net model, but one strategy to optimize a neural net model's weights.

② Back propagation is essentially gradient descent, the most common form of first order optimization techniques. There are many improvements to vanilla back propagation that are being used today, but essentially remains a gradient method.

So, the "forward pass" refers to calculation process, values of the output layers from the input data. It is traversing through all neurons from first to last layer.

And then "backward pass" refers to process of counting changes in weights (de facto learning), using gradient descent algorithm (or similar). Computation is made from last layer, backward to first layer.

NOTE:

① Backward and forward pass makes one "iteration".

② During one iteration, you usually pass a subset of the

data set, which is called "mini-batch".

→ If you pass all the data at once, it is called batch.

③ Epoch means passing the entire data set.

So, 1 epoch = $\frac{\text{Number\_of\_items}}{\text{Batch\_size}}$.

Important derivatives

1. $y = e^x$

So,

$\frac{dy}{dx} = e^x$

② $y = ab$ (Multiplication rule)

So,

$\frac{dy}{dx} = \frac{a\,db}{dx} + \frac{b\,da}{dx}$

③ $y = x^n$ (Power Rule)

So.

$\frac{dy}{dx} = \frac{d(x^n)}{dx} = n\,x^{n-1}$

④ $y = a+b$

So

$\frac{dy}{dx} = \frac{d(a+b)}{dx}$

$= \frac{da}{dx} - \frac{db}{dx}$

5. Difference Rule $y = a-b$

So,

$\frac{dy}{dx} = \frac{d(a-b)}{dx} = \frac{da}{dx} - \frac{db}{dx}$

6. Chain Rule.

$\frac{dy}{dx} = \frac{dy}{du} * \frac{du}{dx}$

$\frac{dy}{dx}$: derivative of y with respect to x.

$\frac{dy}{du}$: derivative of y with respect to u.

$\frac{du}{dx}$: derivative of u with respect to x.

7. $y = \frac{1}{1+e^{-x}}$ (Sigmoid function)

So,

$$\frac{dy}{dx} = \frac{d\left(\frac{1}{1+e^{-x}}\right)}{dx}$$

$$= \frac{d(1+e^{-x})^{-1}}{dx}$$

$$= \frac{d(1+e^{-x})^{-1}}{d(1+e^{-x})} \times \frac{d(1+e^{-x})}{dx} \quad (\because \text{Chain Rule})$$

$$= -1(1+e^{-x})^{-1-1}\left\{\frac{d1}{dx} + \frac{de^{-x}}{dx}\right\}$$

↳ derivative of constant is always 0 (zero)

$$= -1(1+e^{-x})^{-2} \times -1 \times e^{-x}$$

$$= e^{-x} \times (1+e^{-x})^{-2}$$

$$= \frac{e^{-x}}{(1+e^{-x})^2}$$

$$= \frac{1+e^{-x}-1}{(1+e^{-x})^2}$$

$$= \left(\frac{1}{1+e^{-x}}\right)\left(\frac{1+e^{-x}}{1+e^{-x}} - \frac{1}{1+e^{-x}}\right)$$

$$= y(1-y)$$

So, derivative of Sigmoid function

$$\frac{dy}{dx} = y(1-y)$$

Here's how we calculate each steps involved in back propagation from above graph

Now,

$$h_1 = x_1 w_1 + x_2 w_2 + b_1$$
$$= 0.05 * 0.15 + 0.10 * 0.20 + 0.35$$
$$= 0.3775$$

Now we use Sigmoid activation function to get the output of h_1

So, $Out_{h_1} = \frac{1}{1 + e^{-h_1}}$

$$= \frac{1}{1 + e^{-0.3775}}$$
$$= 0.5932$$

Similar process for h_2

$$h_2 = x_1 * w_3 + x_2 * w_4 + b_1$$
$$= 0.05 * 0.25 + 0.10 * 0.30 + 0.35$$
$$= 0.3925$$

$$So, out(h_2) = \frac{1}{1 + e^{-h_2}}$$
$$= \frac{1}{1 + e^{-0.3925}}$$
$$= 0.5968$$

Similarly, we use output from the hidden layer neurons as input for the output layer neurons

$$y_1 = Outh_1 * w_5 + outh_2 * w_6 + b_2$$
$$= 0.5932 * 0.40 + 0.5968 * 0.46 + 0.60$$
$$= 1.1059$$

$$\therefore Out_{y_1} = \frac{1}{1+e^{-y_1}}$$
$$= \frac{1}{1+e^{-1.1059}}$$
$$= 0.7513$$

For Second output layer

$$y_2 = Outh_1 * w_7 + outh_2 * w_8 + b_2$$
$$= 0.5932 * 0.50 + 0.5968 * 0.55 + 0.60$$
$$= 1.2248$$

$$Out_{y_2} = \frac{1}{1+e^{-y_2}}$$
$$= \frac{1}{1+e^{-1.2248}}$$
$$= 0.7729$$

Now, we are using MEAN SQUARE ERROR to get total error.

$$Error(E) = \frac{(y_1(Target) - Out_{y_1})^2}{2} + \frac{(y_2(Target) - Out_{y_2})^2}{2}$$
(y_1 → t_1, y_2 → t_2)

$$= \frac{(0.01 - 0.7513)^2 + (0.99 - 0.7729)^2}{2}$$
$$= 0.2983$$

Now, our goal with back propagation is to update each weight because the target output (t_1) is 0.01 but the neural network predict output as 0.7513 and Similary target (t_2) and out_{y_2}.

Weight update.

$$new(w_5) = old(w_5) - \alpha \frac{d\varepsilon}{dw_5}$$

α → Learning rate (hyper parameter)

ε → total error

↳ Gradient Descent

So,

$$\frac{d\varepsilon}{dw_5} = \frac{d\varepsilon}{dout_{y_1}} \times \frac{dout_{y_1}}{dw_5}$$

$$= \frac{d\varepsilon}{dout_{y_1}} \times \frac{dout_{y_1}}{dy_1} \times \cancel{dw_5} \frac{dy_1}{dw_5}$$

Now we need to figure out each piece in equation.

$$\frac{d\varepsilon}{dout_{y_1}} = \frac{d\left(\frac{(t_1 - out_{y_1})^2 + (t_2 - out_{y_2})^2}{2}\right)}{dout_{y_1}}$$

↳ Constant = 0

$$= \frac{1}{2} \frac{d\{(t_1 - out_{y_1})^2 + (t_2 - out_{y_2})^2\}}{dout_{y_1}}$$

$$= \frac{1}{2} \frac{d(t_1 - out_{y_1})^2}{dout_{y_1}} \quad \left[\because \frac{d(a-b)^2}{db}\right.$$

$$= \frac{1}{2} \times 2(out_{y_1} - t_1) \quad \left[\frac{d(a-b)^2}{d(a-b)} \times \frac{d(a-b)}{db}\right.$$

$$= out_{y_1} - t_1$$

$$\frac{d\,out_{y_1}}{dy_2} = out_{y_1}(1 - out_{y_2})$$

↳ derivative of Sigmoid function.

$$\frac{dy_2}{dw_5} = \frac{d(out_{h_1} * w_5 + out_{h_2} * w_6 + b_2)}{dw_5}$$ → constant

$$= \frac{d(out_{h_1} w_5)}{dw_5}$$

$$= out_{h_1} \frac{dw_5}{dw_5} + w_5 \frac{d\,out_{h_1}}{dw_5}$$ → constant

$$= out_{h_1}$$

So, $\frac{dE}{dw_5} = (out_{y_1} - t_2) * out_{y_2}(1 - out_{y_2}) * out_{h_1}$

$= (0.7513 - 0.01) * 0.7513(1 - 0.7513) * 0.5932$

$= 0.0821$

To update weight, we need to subtract this value from the current weight (multiplied by some learning rate α which is set to 0.5)

But the default value of learning rate is 0.001

So, $new(w_5) = old(w_5) - \alpha \frac{dE}{dw_5}$

$= 0.40 - 0.5 * 0.5932$

$= 0.1034$

$= 0.40 - 0.5 * 0.0821$

$= 0.359$

We can repeat this process to new weights w6.

We know,

$$new(w6) = old(w6) - \alpha \frac{\partial E}{\partial w6}$$

So, $\frac{\partial E}{\partial w6} = \frac{\partial E}{\partial out_{y_1}} \times \frac{\partial out_{y_1}}{\partial w6}$

$$= \frac{\partial E}{\partial out_{y_1}} \times \frac{\partial out_{y_1}}{\partial y_1} \times \frac{\partial y_1}{w6}$$

$$= (out_{y_1} - t_1) * out_{y_1}(1 - out_{y_1}) * out_{h2}$$

$$= (0.7513 - 0.01) * 0.7513(1 - 0.7513) * 0.5968$$

$$= 0.082$$

$$new(w6) = 045 - 0.5 \times 0.082$$

$$= 0.4086$$

Similarly, we can repeat this process to get new weights w7 and w8

$$new(w7) = 0.5113$$

$$new(w8) = 0.5613$$

Note: We perform weights update of hidden layer. Now, we'll continue the backward pass for updating weights w_1, w_2, w_3 and w_4.

Weight update for w_1

$$w_1^{new} = w_1^{old} - \alpha \frac{\partial E}{\partial w_1}$$

So,

$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial h_1} \times \frac{\partial h_1}{\partial w_1}$$

$$= \frac{\partial E}{\partial out_{h_1}} \times \frac{\partial out_{h_1}}{\partial h_1} \times \frac{\partial h_1}{\partial w_1}$$

$$= \frac{\partial E}{\partial out_{y_1}} \times \frac{\partial out_{y_1}}{\partial y_1} \times \frac{\partial y_1}{\partial out_{h_1}} \times \frac{\partial out_{h_1}}{\partial h_1} \times \frac{\partial h_1}{\partial w_1}$$

Now,

$$\frac{\partial E}{\partial out_{y_1}} = \frac{\partial \left\{ \frac{(t_1 - out_{y_1})^2 + (t_2 - out_{y_2})^2}{2} \right\}}{\partial out_{y_1}}$$

$$= \frac{1}{2} \times \frac{\partial (t_1 - out_{y_1})^2}{\partial out_{y_1}}$$

$$= \frac{1}{2} \times 2 * (-1) (t_1 - out_{y_1})$$

$$= out_{y_1} - t_1$$

$$\frac{\partial out_{y_1}}{\partial y_1} = out_{y_1} (1 - out_{y_1})$$

$$\frac{dy_1}{dout_{h_1}} = \frac{d(out_{h_1} * w_5 + out_{h_2} * w_6 + b_2)}{dout_{h_1}}$$ (constant → $out_{h_2} * w_6 + b_2$ is constant)

$$= w_5 \frac{dout_{h_1}}{dout_{h_1}} + out_{h_1} \times \frac{dw_5}{dout_{h_1}}$$

$$= w_5$$

$$\frac{dout_{h_1}}{dh_1} = out_{h_1}(1 - out_{h_1})$$

$$\frac{dh_1}{dw_1} = \frac{d(x_1 w_1 + x_2 w_2 + b_1)}{dw_1}$$ ($x_2 w_2 + b_1$ → constant)

$$= x_1 \frac{dw_1}{dw_1} + w_1 \frac{dx_1}{dw_1}$$

$$= x_1$$

$$\text{So, } \frac{dE}{dw_1} = (out_{y_1} - t_1) * out_{y_2}(1 - out_{y_1}) * w_5 * out_{h_1}(1 - out_{h_1}) * x_1$$

$$= (0.7513 - 0.01) * 0.7513(1 - 0.7513) * 0.40 * 0.5932(1 - 0.5932) * 0.05$$

$$= 0.00066$$

Now,

$$new(w_1) = (w_1)_{old} - \alpha \frac{dE}{dw_1}$$

$$= 0.15 - 0.5 * 0.00066$$

$$= 0.14967$$

We can repeat this process to new weights w_2.

We know

$$\text{New}(w_2) = \text{old}(w_2) - \alpha \frac{\partial E}{\partial w_2}.$$

$$\frac{\partial E}{\partial w_2} = \frac{\partial E}{\partial h_1} \times \frac{\partial h_1}{\partial w_2}.$$

$$= \frac{\partial E}{\partial \text{out}_{h_1}} \times \frac{\partial \text{out}_{h_1}}{\partial h_1} \times \frac{\partial h_1}{\partial w_2}$$

$$= \frac{\partial E}{\partial y_1} \times \frac{\partial y_1}{\partial \text{out}_{h_1}} \times \frac{\partial \text{out}_{h_1}}{\partial h_1} \times \frac{\partial h_1}{\partial w_2}.$$

$$= \frac{\partial E}{\partial \text{out}_{y_1}} \times \frac{\partial \text{out}_{y_1}}{\partial y_1} \times \frac{\partial y_1}{\partial \text{out}_{h_1}} \times \frac{\partial \text{out}_{h_1}}{\partial h_1} \times \frac{\partial h_1}{\partial w_2}.$$

$$= (\text{out}_{y_1} - t_1) * \text{out}_{y_1}(1 - \text{out}_{y_1}) * w_5 * \text{out}_{h_1}(1 - \text{out}_{h_1}) * x_2.$$

(Similarly to $\partial E / \partial w_1$)

$$= (0.7513 - 0.01) * 0.7513\,(1 - 0.7513) * 0.40 * 0.5932(1 - 0.5932) * 0.10$$

$$= 0.0013$$

So, $\text{new}(w_2) = \text{old}(w_2) - \alpha \frac{\partial E}{\partial w_2}$

$$= 0.20 - 0.5 * 0.0013$$

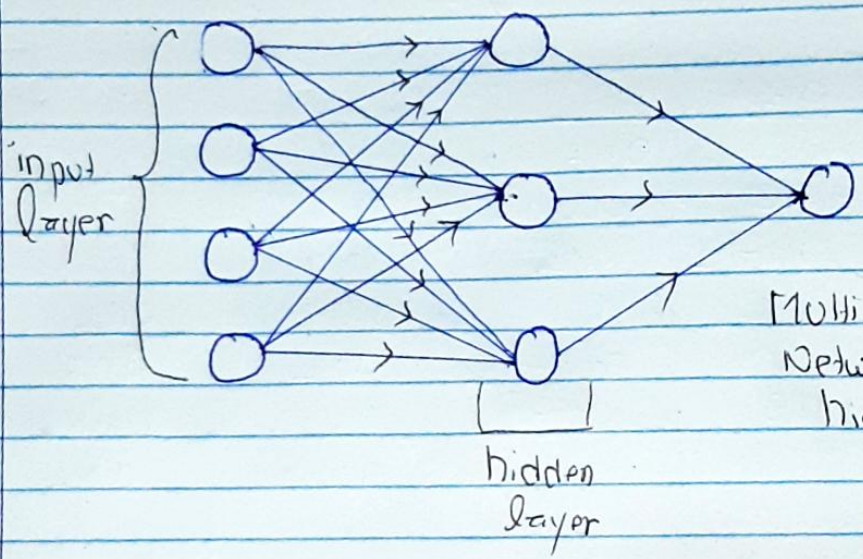
$$= 0.19935$$

Repeating this for w_3 and w_4.

$\therefore$ $\text{New}(w_3) = 0.2497$ and $\text{new}(w_4) = 0.29950$

1. Why don't we initialize the weights of neural network to zero?

→ Let's consider a neural network with 1 hidden layer.



Multi layer Neural Network with 1 hidden layer.

Lets say that each node in the hidden layer computes the activation function 'a-h' defined by:

$$z = w_h * x + b_h$$

$$a_h = \text{Sigmoid}(z)$$

Where

w_h = Weights of the hidden layer to be updated in each pass.

x = input from the previous input layer.

b_h = bias from the hidden layer.

Now, if you initialize Wl and b of all nodes as zero, all nodes of the hidden layer will have same weights 'Wl' and same bias 'b'. Hence, they will have same activation functions.

The output layer will compute

$$\hat{y}_o = \text{Sigmoid}(W_o * a_h + b_o)$$

Since Wout and bout are initialized to zero as well (According to the questions).

$$\hat{y}_{out} = \text{Sigmoid}(0) \text{ which is equal to } 0.5$$

This ends forward propagation.

During back propagation, we compute dWl, db, and da starting from loss function L.

There is the bit of calculus involved but it turns out that dWl will be same for all nodes of the hidden layer since Z is the same for all nodes.

So, when we update the weights using the below formula, Wl is the same for all three nodes.

$$W_l = W_l - (\underbrace{\text{alpha}}_{\text{learning rate}} * dW)$$

So all of your nodes will end up with the same value and it will be no different that having

just one node in the hidden layer. Because of this, your network will end up learning just one function ~~on~~ while the goal of a neural network is to have different nodes computes different function. This is called 'Symmetry problem'.

In order to break this symmetry, we initialize the weights randomly.

"HAPPY MATHS, HAPPY AI"