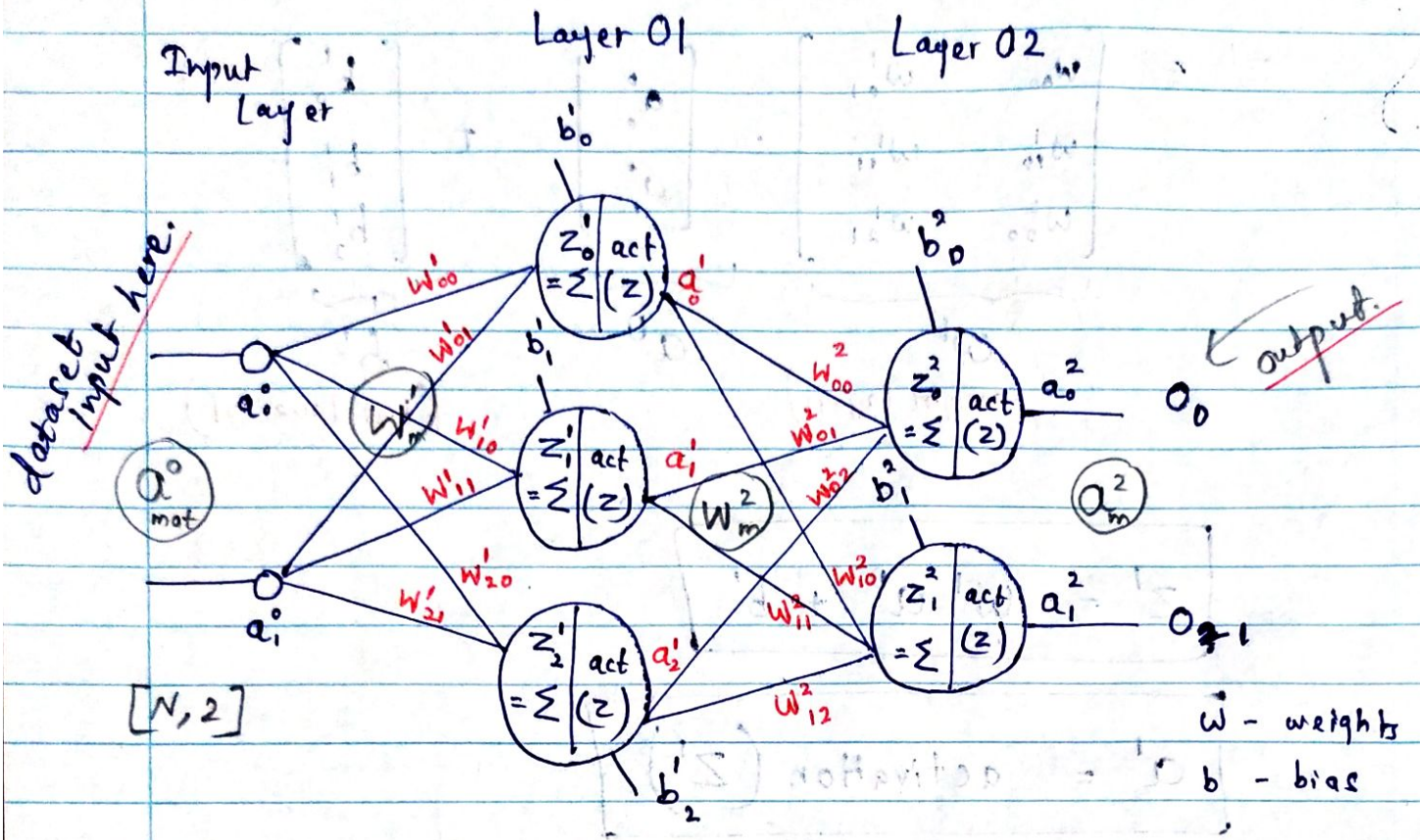
Mathematic of neural network.

No: _____ Date: __/__/____



$\dot{w}$ - weights

b - bias

* act( ) – activation function (ReLU, softmax...) the purpose of the activation function is the introduce the non-linearity to the network. Other wise, the node will be just a linear representation of previous layer.

Layer 01 – (Forward prop.)

$$z'_0 = w'_{00}a^0_0 + w'_{01}a^0_1 + b'_0 \qquad a'_0 = act(z'_0)$$

$$z'_1 = w'_{10}a^0_0 + w'_{11}a^0_1 + b'_1 \qquad a'_1 = act(z'_1)$$

$$z'_2 = w'_{20}a^0_0 + w'_{21}a^0_1 + b'_2 \qquad a'_2 = act(z'_2)$$

$$z' = \begin{bmatrix} w'_{00} & w'_{01} \\ w'_{10} & w'_{11} \\ w'_{20} & w'_{21} \end{bmatrix} \cdot \begin{bmatrix} a^0_0 \\ a^0_1 \end{bmatrix} + \begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \end{bmatrix}$$

dot product

3×2 $\quad$ 2×1 $\quad$ 3×1

W' (mat) $\quad$ a^0 (mat) $\quad$ Also

$$z^1 = \begin{bmatrix} w_{00}^1 & w_{01}^1 \\ w_{10}^1 & w_{11}^1 \\ w_{20}^2 & w_{21}^1 \end{bmatrix} \cdot \begin{bmatrix} a_0^0 \\ a_1^0 \end{bmatrix} + \begin{bmatrix} b_0^1 \\ b_1^1 \\ b_2^1 \end{bmatrix}$$

w^1 (weight layer1) $\quad a^0$ $\quad b^1$ (bias layer'01)

$$\boxed{z^1 = w^1 a^0 + b^1}$$

$$\boxed{a^1 = \text{activation}(z^1)}$$

Same thing happens in the each and every layer in the neural network, so that we can represent the process of the forward propagation of the 2 layer ANN as follows,

Forward Prop.

$$\boxed{\begin{aligned} z^1 &= w^1 a^0 + b^1 \\ a^1 &= act^1(z^1) \\ z^2 &= w^2 a^1 + b^2 \\ a^2 &= act^2(z^2) \end{aligned}}$$

output of layer 1 → $a^1 = act^1(z^1)$

output of layer 2 (output) → $a^2 = act^2(z^2)$

a^0 w^1 b^1 a^1 w^2 b^2 a^2 ← matrices.

(2×1) (3×2) (3×1) (3×1) (2×3) (2×1) (2×1)

No: _____ Date: __/__/____

Backward prop.

→ In the backward pass, the gradient of the loss with respect to the each weight in the networks in the networks is computed

→ This is done by the applying the chain rule of the calculus to calculate the partial derivatives of the loss with respect to to the weights and biases of the networks.

→ The gradients indicate how much the loss would increases or decrease if the corresponding weights and biases were adjusted.

→ Since the chain rule is important,

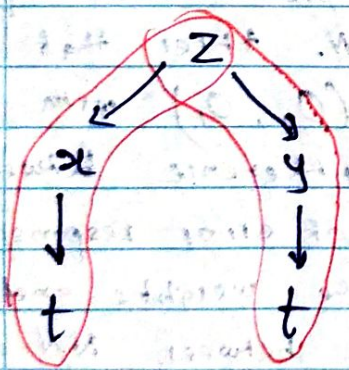
→ example for chain rule in partial derivative. (EX01)

$z = x^3y + 3x^2 + y^2$ / $x = 4 + t^2$ / $y = 5t^3$

Find the $\frac{dz}{dt}$

d - derivative

∂ - partial derivative.

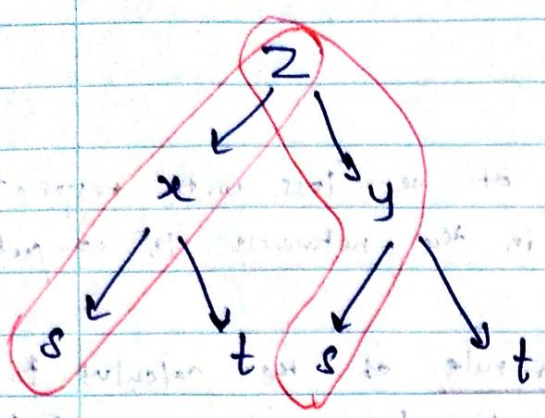


$$\frac{dz}{dt} = \left(\frac{\partial z}{\partial x} \cdot \frac{dx}{dt}\right) + \left(\frac{\partial z}{\partial y} \cdot \frac{dy}{dt}\right)$$

→ example for chain rule in partial derivative (EX02).

$z = 4x^3y^5$ / $x = 3s^2t$ / $y = 2s^3t^2$

Find, $\frac{\partial z}{\partial s}$



$$\frac{\partial z}{\partial s} = \left(\frac{\partial x}{\partial s} \cdot \frac{\partial z}{\partial x}\right)\left(\frac{\partial y}{\partial s} \cdot \frac{\partial z}{\partial y}\right)$$

The chain rule of the partial derivative is being used in the backword pass of the NN training process.

Again to the backward pass process...

In a training dataset, there are two parts.

Training dataset

- Input data (x)
- target outcome / Labels (y)

In the forward pass, we let the NN to predict the output by passing the Input data through the NN. After that we compares the output of the NN (O_0, O_1) with the target outcome / labels and find the difference between them. After that we calculate the value of error responsible for each weigh and bias and update the weights and bias in order to minimize the difference between NN out and target outcomes.

using "cost function" we find the overall error made by the NN. cost function give a quantative measure of the error made by the Neural network.

No: _____ Date: __/__/____

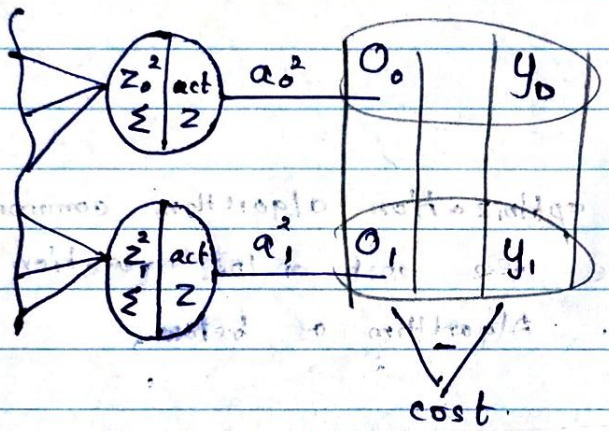
Cost function.

cost function give a quantative measures of all the error made by the neural network. There are different types of cost functions used in different ANN problems,

- MSE – mean square error. – used in this example.
- Binary crossentroly
- Hinge loss.

$$MSE = \frac{1}{N} \sum_{i=0}^{N-1} (O_i - y_i)$$

Layer02



O – output of ANN
y – target output

target / labels

→ For our example NN, the cost function C(x) will be,

$$C() = (O_0 - y_0)^2 + (O_1 - y_1)^2$$ (since y is not changing it, is a constant.)

since the "y" is constant, the C(x) is a function O.

$C(O_i)$ and also, the O is ~~depence~~ depent on the values of parameters (weight and bias)

Cost is function of all weight and bias.

$$C(w_{00}^1, w_{01}^1, w_{10}^1, \ldots w_{12}^2, b_1^2) \rightarrow C(P_i)$$

parameters of NN (P_i) ← any parameter.

[we use P_i to represent all parameter is ANN such as w_{00}^1, w_{01}^1 b_1^2 ...]

Now we know that cost is the function of NN's parameter such and all weights and biases.

$$C\underbrace{(w'_{00}, w'_{01} \ldots w^2_{12}, b^2_1)}_{P_i} \longrightarrow C(P_i)$$

(→) the main purpose of ANN training process is to minimize the cost or loss function during the training the model. To minimize the cost, we use the Gradient Descent. algorithm.

Gradient Deccent.

Gradient Descent is a optimization algorithm commonly used in deeplearning to minimize the cost or loss function during the training the model. Algorithm of below,

learning rate.

updated parameter → $P_{i_{new}} = P_{i_{old}} - \eta \frac{\partial C}{\partial P_i}$

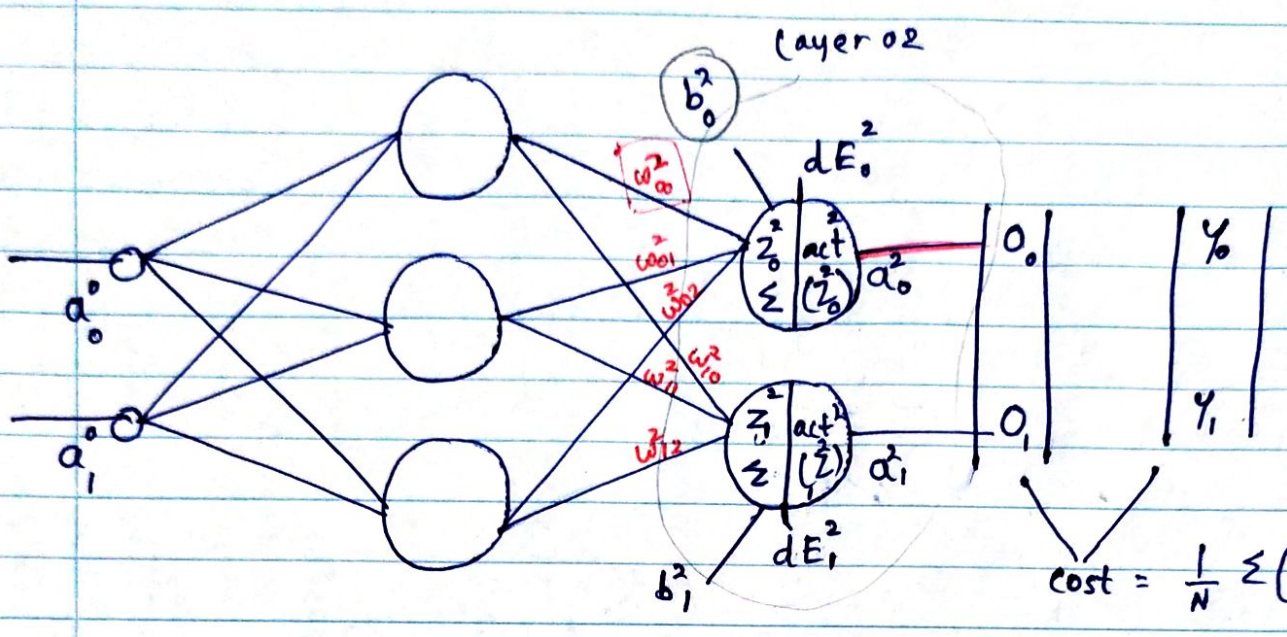
η – learning rate.

$\frac{\partial C}{\partial P_i}$ = the slope of cost function under respect particular parameter. (P_i) (partial derivative.)

P_i is each and every parameter of the ANN. for example for 'w_{00}'' of first layer of the ANN.

$$w'_{00\,(new)} = w'_{00\,(old)} - \eta \frac{\partial C}{\partial w'_{00}}$$

→ Let's assume the layer 02 of the neural network.



→ Let's assume the b_0^2 bias of the layer 02, first neuron.

$$b_{0\,(new)}^2 = b_{0\,(old)}^2 - \eta \frac{\partial C}{\partial b_0^2}$$

← we cannot find this straigword. we have to use the partial derivative chain rule to find this.

$$\frac{\partial C}{\partial b_0^2} = \frac{\partial C}{\partial O_0} \cdot \frac{\partial O_0}{\partial Z_0^2} \cdot \frac{\partial Z_0^2}{\partial b_0^2} \quad \text{— "A" equation.}$$

this $\left(\frac{\partial C}{\partial O_0} \cdot \frac{\partial O^0}{\partial Z_0^2}\right)$ is common to the all the parameters of the particular neuron, so that it can be represent as "$d E^2$" error factor.

discuss more in the parts that coming up.

Cost → Out → a → z — b^2

chain rule

each neuron of the ANN has seperate "dE_0^2" value.

for example, this is common.

$$\frac{dC}{\partial b_0^2} = dE_0^2 \cdot \frac{\partial z_0^2}{\partial b_0^2} \qquad \frac{\partial C}{\partial w_{00}^2} = dE_0^2 \cdot \frac{\partial z_0^2}{\partial w_{00}^2}$$

$$\frac{\partial C}{\partial b_0^2} = dE^2 \cdot \frac{dz_0^2}{\partial b_0^2}$$

(eq01) $$dE_0^2 = \frac{\partial C}{\partial O_0} \cdot \frac{\partial O^2}{\partial z_0^2} \quad \text{—(1)}$$ ← This is common for particular neoron.

~~$dE_1^2 = (O_0 - y_0)^2$~~

$$\frac{\partial C}{\partial O_0} = \frac{\partial \Sigma (O_i - y_i)^2}{\partial O_0} \rightarrow \frac{\partial \left[(O_0 - y_0)^2 + (O_1 - y_1)^2 \right] \times 1/2}{\partial O_0}$$

(eq02) $$\frac{\partial C}{\partial O_0} = (O_0 - y_0) \quad \text{—(2)}$$

(eq03) $$\frac{\partial O_0}{\partial Z_0} = act_2'(Z_0^2) \quad \text{—(3)}$$ derivative of the activation function used in the particular layer. Act_2 — activation funct of layer 2.]

using eq01, eq02 and eq03 .

$$\boxed{dE_0^2 = (O_0 - y_0) \cdot act_2'(Z_0^2)}$$ derivative of the activation function.

above given the error vector for the layer 02, 0th node/neron.

For the layer 02, 1st node,

$$\boxed{dE_1^2 = (O_1 - y_1) \cdot act_2'(Z_0^2)}$$

And also, error vector for the layer 2 can be represent using a matrix as below,

$$\boxed{dE^2 = \begin{bmatrix} dE_0^2 \\ dE_1^2 \end{bmatrix}}$$

Since we know the (dE_0^2) and (dE_1^2) now, we can find the ~~loss for~~ $\frac{\partial C}{\partial b_0^2}$ and all the parameters (ω and b) at Layer 02.

As a example let find the b_0^2 and ω_{00}^2.

→ $\frac{\partial C}{\partial b_0^2}$

$$\frac{\partial C}{\partial b_0^2} = dE_0^2 \cdot \frac{\partial z_0^2}{\partial b_0^2}$$ ← (using equation given in the 7 page)

$$dE_0^2 = (O_0 - y_0) \cdot act_2'(z_0^2)$$

$$\frac{\partial z^2}{\partial b_0^2} = \frac{\partial (\omega_{00}^2 a_0' + \omega_{01}^2 a_1' + \omega_{02} a_2' + b_0^2)}{\partial b_0^2}$$ ← partial derivative.

$$\frac{\partial z^2}{\partial b_0^2} = 1$$

so that, $$\boxed{\frac{\partial C}{\partial b_0^2} = \underbrace{(O_0 - y_0) \cdot act_2'(z_0^2)}_{dE_0^2}}$$

→ $\frac{\partial C}{\partial \omega_{00}^2}$

$$\frac{\partial C}{\partial \omega_{00}^2} = dE_0^2 \cdot \frac{\partial z_0^2}{\partial b_0^2} \frac{\partial}{\partial \omega_{00}^2}$$

$$\frac{\partial z^2}{\partial \omega_{00}^2 \cancel{\partial b_0^2}} = \frac{\partial (\omega_{00}^2 a_0' + \omega_{01}^2 a_1' + \omega_{02} a_2' + b_0^2)}{\cancel{\partial b_0^2}\ \partial \omega_{00}^2}$$

$$\frac{\partial z^2}{\partial \omega_{00}^2} = a_0'$$

so that,

~~So that~~, ~~$\frac{\partial C}{\partial \omega_{00}^2}$~~ ~~$\delta_0^+$~~ $$\boxed{\frac{\partial C}{\partial \omega_{00}^2} = \underbrace{(O_0 - y_0) \cdot act_2'(z_0^2)}_{dE_0^2} \cdot a_0'}$$

In a similar way, we can calculate the each and every parameter errors in the layer 2. (1st layer of the backward pass / last layer).

→ below I have given the error calculations for each and every weight parameter.

$\frac{\partial C}{\partial w_{00}^2} = \partial E_0^2 \cdot a_0^1$	$\frac{\partial C}{\partial w_{01}^2} = \partial E_0^2 \cdot a_1^1$	$\frac{\partial C}{\partial w_{02}^2} = \partial E_0^2 \cdot a_2^1$
$\frac{\partial C}{\partial w_{10}^2} = \partial E_1^2 \cdot a_0^1$	$\frac{\partial C}{\partial w_{11}^2} = \partial E_1^2 \cdot a_1^1$	$\frac{\partial C}{\partial w_{12}^2} = \partial E_1^2 \cdot a_2^1$

→ As you can remember, our ultimate goal is to, below, (for weights)

~~$w_{i_{new}} = w_{i_{old}} - \eta \frac{\partial C}{\partial P_i}$~~

$$P_{i_{new}} = P_{i_{old}} - \eta \frac{\partial C}{\partial P_i} \longrightarrow \text{for weights} \longrightarrow \boxed{w_{i_{new}} = w_{i_{old}} - \eta \frac{\partial C}{\partial w_i}}$$

As you can remember, we the weights of the last layer with a 2×3 matrix. Likwise, we have ~~represent~~ build the 2×3 matrix for $\left(\eta \frac{\partial C}{\partial w}\right)$ this as well. Then only we can substract.

$$\partial w^2 = \begin{bmatrix} \partial E_0^2 a_0^1 & \partial E_0^2 a_1^1 & \partial E_0^2 a_2^1 \\ \partial E_1^2 a_0^1 & \partial E_1^2 a_1^1 & \partial E_1^2 a_2^1 \end{bmatrix}$$

$$\partial w^2 = \begin{bmatrix} \partial E_0^2 \\ \partial E_1^2 \end{bmatrix} \cdot \begin{bmatrix} a_0^1 & a_1^1 & a_2^1 \end{bmatrix}$$

No: Date: __/__/__

$$\underset{2\times3}{\partial w^2} = \underset{3\times1}{\partial E^2} \cdot \underset{1\times2}{(a^1)^T}$$

← For weights. Layer 02

$$\partial b^2 = \partial E^2$$

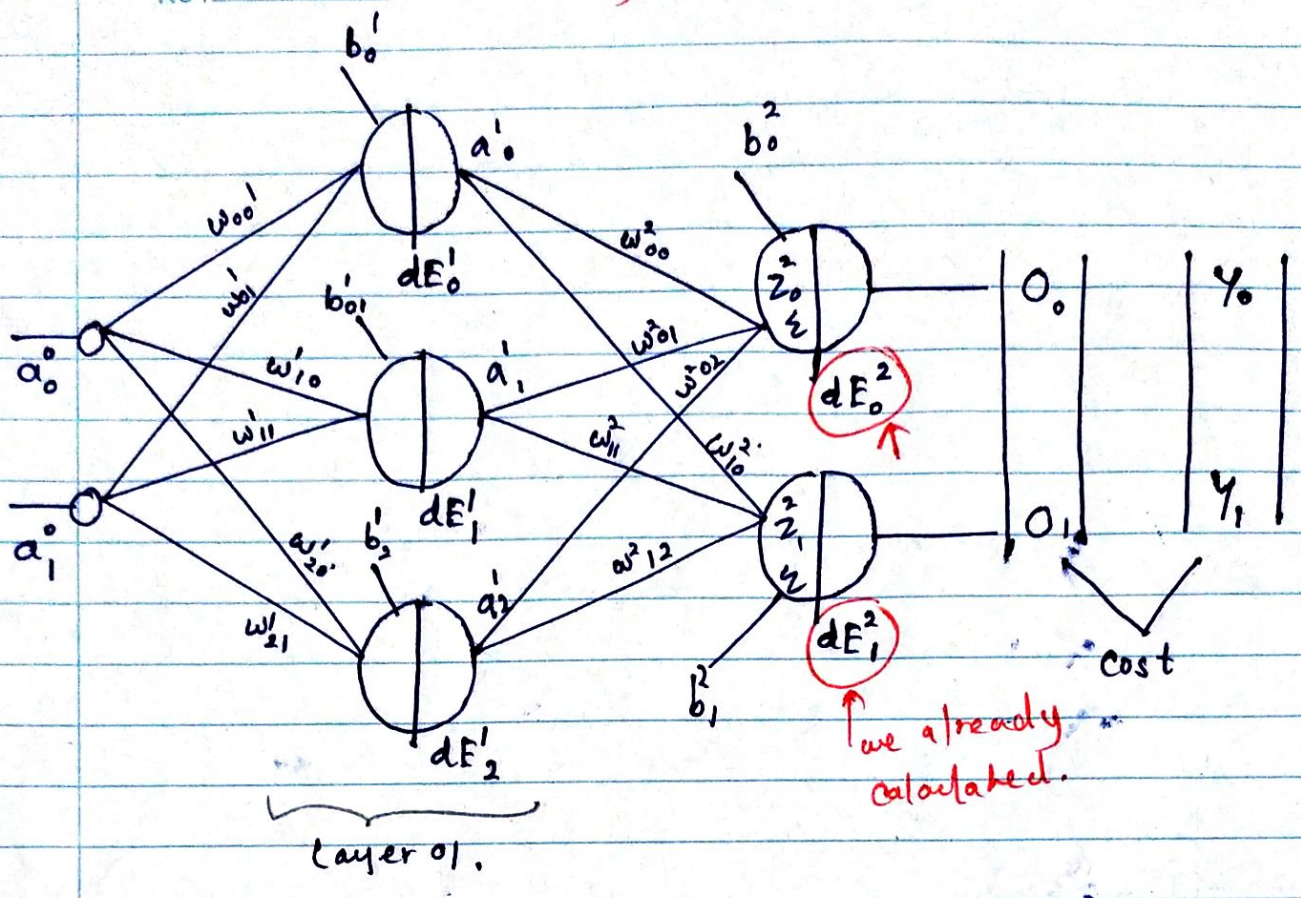
← For bias of Layer 02

So that we can run the gradient Decent algorithm to update the weight and bias for layer 02 (last layer).

$$W^2_{new} = W^2_{old} - \eta \, \partial w^2$$

$$b^2_{new} = b^2_{old} - \eta \, \partial b^2$$

now we are done with the layer 02, Now we have to do the similar thing to the layer 01 al well.



→ we know that we are going through the backward pass of the neural network, so that we already have known the dE_0^2 and dE_1^2.

→ let's assume the "b_0^1" bias of the layer 01, first neuron.

$$b_0^1 \underset{\text{(new)}}{=} b_0^1 \underset{\text{(old)}}{-} \eta \frac{\partial C}{\partial b_0^1}$$ ← like we did before, we have to find this using the chain rule of partial derivative.

$b_0^1 \rightarrow z_0^1 \rightarrow a_0^1 \longrightarrow z_0^2 \rightarrow a_0^2 \rightarrow$ cost.

$a_0^1 \rightarrow z_1^2 \rightarrow a_1^2 \rightarrow$ cost

chain rule.

$$\frac{\partial C}{\partial b_0^1} = \left[\underbrace{\frac{\partial C}{\partial a_0} \frac{\partial O_0}{\partial z_0^2}}_{dE_0^2} \frac{\partial z_0^2}{\partial a_0^1} + \underbrace{\frac{\partial C}{\partial O_1} \frac{\partial O_1}{\partial z_1^2}}_{dE_1^2} \frac{\partial z_1^2}{\partial a_0^1} \right] \cdot \frac{\partial a_0^1}{\partial z_0^1} \cdot \frac{\partial z_0^1}{\partial b_0^1}$$

we derived these two in the layer 02 process.

No: Date: __/__/____

$$\frac{\partial C}{\partial b_0^1} = \left[dE_0^2 \cdot \frac{\partial z_0^2}{\partial a_0^1} + \partial E_1^2 \cdot \frac{\partial z_1^2}{\partial a_0^1} \right] \frac{\partial a_0^1}{\partial z_0^1} \cdot \frac{\partial z_0^1}{\partial b_0^1}$$

dE_0^1

First we should find the $\rightarrow \frac{\partial C}{\partial a_0^1}$

$$\frac{\partial C}{\partial a_0^1} = \left[dE_0^2 \cdot \frac{\partial z_0^2}{\partial a_0^1} + \partial E_1^2 \cdot \frac{\partial z_1^2}{\partial a_0^1} \right]$$

dE_0^1 ← error vector.

$$\frac{\partial z_0^2}{\partial a_0^1} = \frac{\partial \left(w_{00}^2 a_0^1 + w^2 a_1^1 + w^2 a_2^1 + b_0^2 \right)}{\partial a_0^1}$$

$$\frac{\partial z_0^2}{\partial a_0^1} = a_0^1 \qquad \boxed{\frac{\partial z_0^2}{\partial a_0^1} = w_{00}^2}$$

Likewise, $\boxed{\frac{\partial z_1^2}{\partial a_0^1} = w_{10}^2}$

$$\therefore \boxed{\frac{\partial C}{\partial a_0^1} = \partial E_0^2 \cdot w_{00}^2 + \partial E_1^2 \cdot w_{10}^2}$$

Likewise we can find the $\left(\frac{\partial C}{\partial a_0^1}\right), \left(\frac{\partial C}{\partial a_1^1}\right), \left(\frac{\partial C}{\partial a_2^1}\right)$ for each node layer 01.

$$\frac{\partial C}{\partial a_0^1} = \partial E_0^2 \cdot w_{00}^2 + \partial E_1^2 \cdot w_{10}^2$$

$$\frac{\partial C}{\partial a_1^1} = \partial E_0^2 \cdot w_{01}^2 + \partial E_1^2 \cdot w_{11}^2$$

$$\frac{\partial C}{\partial a_2^1} = \partial E_0^2 \cdot w_{02}^2 + \partial E_1^2 \cdot w_{12}^2$$

← matrix.

$$\begin{bmatrix} w_{00}^2 \, \partial E_0^2 + w_{10}^2 \, \partial E_1^2 \\ w_{01}^2 \, \partial E_0^2 + w_{11}^2 \, \partial E_1^2 \\ w_{02}^2 \, \partial E_0^2 + w_{12}^2 \, \partial E_1^2 \end{bmatrix} = \underbrace{\begin{bmatrix} w_{00}^2 & w_{10}^2 \\ w_{01}^2 & w_{11}^2 \\ w_{02}^2 & w_{12}^2 \end{bmatrix}}_{[w^2]^T} \cdot \begin{bmatrix} \partial E_0^2 \\ \partial E_1^2 \end{bmatrix}$$

$$\boxed{\partial a^1 = [w^2]^T \, \partial E^2}$$

As the equation we build in the previous page,

$$\frac{\partial c}{\partial b_0^1} = \left[\partial E_0^2 \cdot \frac{\partial z a^2}{\partial a_0^1} + \partial E_1^2 \cdot \frac{\partial z_1^2}{\partial a_0^1} \right] \cdot \frac{\partial a_0^1}{\partial z_0^1} \cdot \frac{\partial z_0^1}{\partial b^1},$$

$$\frac{\partial c}{\partial b_0^1} = \underbrace{\left[\frac{\partial c}{\partial a_0^1} \right] \cdot \frac{\partial a_0^1}{\partial z_0^1}}_{\partial E_0^1} \cdot \underbrace{\frac{\partial z_0^1}{\partial b^1}}_{1}$$

derivation of the activation function used in the layer 01

$$\boxed{\frac{\partial c}{\partial b_0^1} = \partial E_0^1 = \partial a_0^1 \cdot act_1'(z_0^1)} \leftarrow \text{for bias of layer 01}$$

~~As matrix, we can represent for the error for the bias using below equation,~~

$$\boxed{\partial b^1 = \partial E^1}$$

error matrix for layer 01

$$\underset{3\times 1}{\partial E^1} = \partial a^1 \cdot act_1' \cdot \begin{bmatrix} z_0^1 \\ z_1^1 \\ z_2^1 \end{bmatrix}$$

$$\boxed{\partial E^1 = \partial a^1 \cdot act'(z^1)}$$

For bias in Layer 01

$$\frac{\partial C}{\partial b_0^1} = \partial E_0^1 \cdot \underbrace{\frac{\partial z_0^1}{\partial b_0^1}}_{1} = \partial E_0^1$$

$$\frac{\partial C}{\partial b_1^1} = \partial E_1^1$$

$$\frac{\partial C}{\partial b_2^1} = \partial E_2^1$$

matrix.

$$\underset{3\times 1}{\partial b^1} = \underset{3\times 1}{\partial E^1}$$

For weight in Layer 0.2

$$\partial w^1 = \begin{bmatrix} \frac{\partial C}{\partial w_{00}^1} & \frac{\partial C}{\partial w_{01}^1} \\ \frac{\partial C}{\partial w_{10}^1} & \frac{\partial C}{\partial w_{11}^1} \\ \frac{\partial C}{\partial w_{20}^1} & \frac{\partial C}{\partial w_{21}^1} \end{bmatrix} = \begin{bmatrix} \partial E_0^1 \cdot \frac{\partial z_0^1}{\partial w_{00}^1} & \partial E_0^1 \cdot \frac{\partial z_0^1}{\partial w_{01}^1} \\ \partial E_1^1 \cdot \frac{\partial z_1^1}{\partial w_{10}^1} & \partial E_1^1 \frac{\partial z_1^1}{\partial w_{11}^1} \\ \partial E_2^1 \frac{\partial z_2^1}{\partial w_{20}^1} & \partial E_2^1 \cdot \frac{\partial z_2^1}{\partial w_{21}^1} \end{bmatrix}$$

3×2

$$\frac{\partial z_0^1}{\partial w_{00}^1} = \frac{\partial (w_{00}^1 a_0^0 + w_{01}^1 \cdot a_1^0 + b_0^1)}{\partial w_{00}^1} = a_0^0$$

$$\frac{\partial z_0^1}{\partial w_{00}^1} = a_0^0$$

No :

Date: __/__/____

$$\frac{\partial z'_0}{\partial w'_{00}} = a^0_0 \qquad \frac{\partial z'_0}{\partial w'_{01}} = a^0_1$$

$$\frac{\partial z'_1}{\partial w'_{10}} = a^0_0 \qquad \frac{\partial z'_1}{\partial w'_{11}} = a^0_1$$

$$\frac{\partial z'_2}{\partial w'_{20}} = a^0_0 \qquad \frac{\partial z'_2}{\partial w'_{21}} = a^0_1$$

$$\therefore \partial w' = \begin{bmatrix} \partial E'_0 \cdot a^0_0 & \partial E'_0 a^0_1 \\ \partial E'_1 a^0_0 & \partial E'_1 a^0_1 \\ \partial E'_2 a^0_0 & \partial E'_2 a^0_1 \end{bmatrix}$$

$$= \begin{bmatrix} \partial E'_0 \\ \partial E'_1 \\ \partial E'_2 \end{bmatrix} \cdot \begin{bmatrix} a^0_0 & a^0_1 \end{bmatrix}$$

$3 \times 1 \qquad 1 \times 2$

matrix

$$\underset{3\times 2}{\partial w'} = \underset{3\times 1}{\partial E'} \cdot \underset{1\times 2}{[a^0]^T}$$

← For weight at layer 01

$$\partial b' = \partial E'$$

← For bias of layer 01 (previously found)

so that we can run the gradient Decent algorithm to update the weights and bias for layer 01 (First layer).

$$\underset{new}{w^1} = \underset{old}{w^1} - \eta \partial w^1$$

$$\underset{new}{b^1} = \underset{old}{b^1} - \eta \partial b^1$$

Now we are done with the layer 01, ~~now we have to do the similar thing to the~~ and layer 02 as well.

Now we will summerize the entire process.

No: Date: / /

Summerize the entire training process.

→ Forward Pass

$$z^1 = w^1 a^0 + b^1$$

$$a^1 = act^1(z^1)$$

$$z^2 = w^2 a^1 + b^2$$

$$a^2 = act^2(z^2)$$

} calculated. page 02

→ Backward Propagation

1) compute gradients. (sequence is given below)

$$\partial E^2 = (o - y)\, act_2'(z^2)$$ page 08

→ $$\partial b^2 = \partial E^2$$ page 11

→ $$\partial w^2 = \partial E^2 \cdot [a^1]^T$$ page 11

$$\partial a^1 = [w^2]^T \cdot \partial E^2$$ page 14

$$\partial E^1 = \partial a^1 \cdot act_1'(z^1)$$ page 15

→ $$\partial b^1 = \partial E^1$$ page 15

→ $$\partial w^1 = \partial E^1 \cdot [a^0]^T$$ page 16

2) update weights and bias.

$$w^1 = w^1 - \eta \times \partial w^1$$

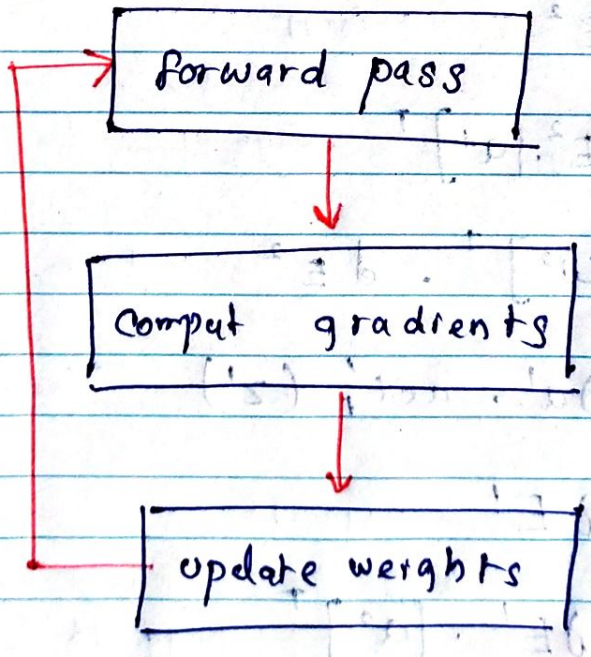
$$b^1 = b^1 - \eta \times \partial b^1$$

$$w^2 = w^2 - \eta \times \partial w^2$$

$$b^2 = b^2 - \eta \times \partial b^2$$

η is learning rate that should be defined by the programer. Usually it is 0.01 or 0.1 (constant).

Flow diagram



~~one cycle is one epoch.~~