DATE

Neural Network :-

Prerequisite :-

Chain of derivative :-

$x \nearrow$ z $\searrow y$ $\Rightarrow z$ is dependent on x & y

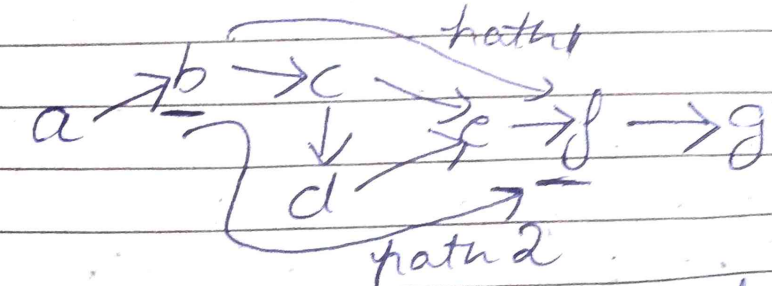
i.e $z = f(x, y)$

x, y both appear in z

$x \to y \to z \Rightarrow$ y dependent on x and z dependent on y (only y appear in z)

→ Derivative of a var wrt another var is sum of product of derivative along each path

eg:-



$$\frac{\partial f}{\partial b} = \underbrace{\frac{\partial f}{\partial e}\frac{\partial e}{\partial c}\frac{\partial c}{\partial b}}_{\text{path 1}} + \underbrace{\frac{\partial f}{\partial e}\frac{\partial e}{\partial d}\frac{\partial d}{\partial c}\frac{\partial c}{\partial b}}_{\text{path 2}}$$

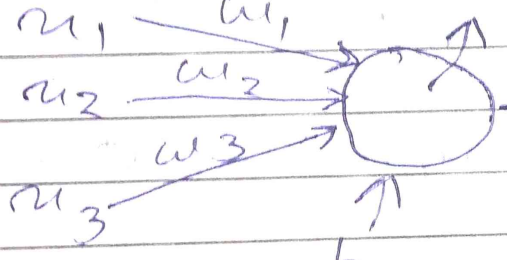
eg $t \rightarrow x = f(t)$, $t \rightarrow y = g(t)$, $x, y \rightarrow z = h(x, y)$

$$\frac{\partial z}{\partial t} = \frac{\partial z}{\partial x}\frac{dx}{dt} + \frac{\partial z}{\partial y}\frac{dy}{dt}$$

↳ x, y only dependent on t

Neuron:

$$z = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$



y = output
= activation (z)
= $\sigma(z)$
↳ goes to input in next layer

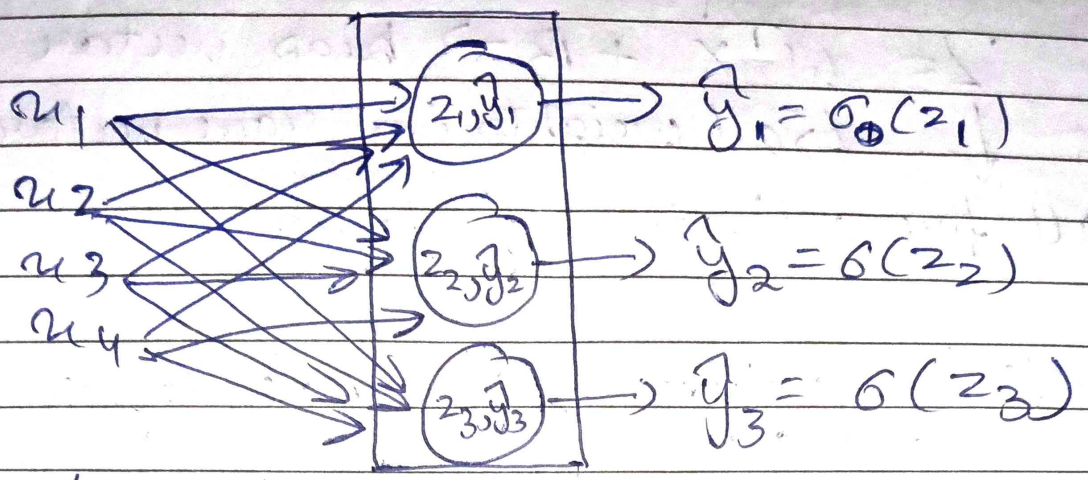
Matrix form: $w = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}$ $X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$

$$z = \underbrace{w^T x} + b$$

↳ dot product ($\vec{w} \cdot \vec{x}$)
↳ helps in vectorization

→ output calculated by passing to $\sigma(z)$.

layer: Multiple neuron in a layer are fed same input.



b_i = bias of ith neuron
W_i = weight matrix of ith ~~neuron~~ layer
z_i = pre activation of ith neuron
y_i = prediction " " "
x = input vector (given to all)

→ activation function σ is same for every neuron in layer

Matrix form :-

$$z_1 = w_1^T x + b_1$$
$$z_2 = w_2^T x + b_2$$
$$z_3 = w_3^T x + b_3$$

$$Z = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} \leftarrow w_1^T \rightarrow \\ \leftarrow w_2^T \rightarrow \\ \leftarrow w_3^T \rightarrow \end{bmatrix} X + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

$$= \begin{bmatrix} \uparrow & \uparrow & \uparrow \\ w_1 & w_2 & w_3 \\ \downarrow & \downarrow & \downarrow \end{bmatrix}^T X + B$$

$$= W^T X + B$$

→ assume weight of neurons are
(#) column vectors

DATE

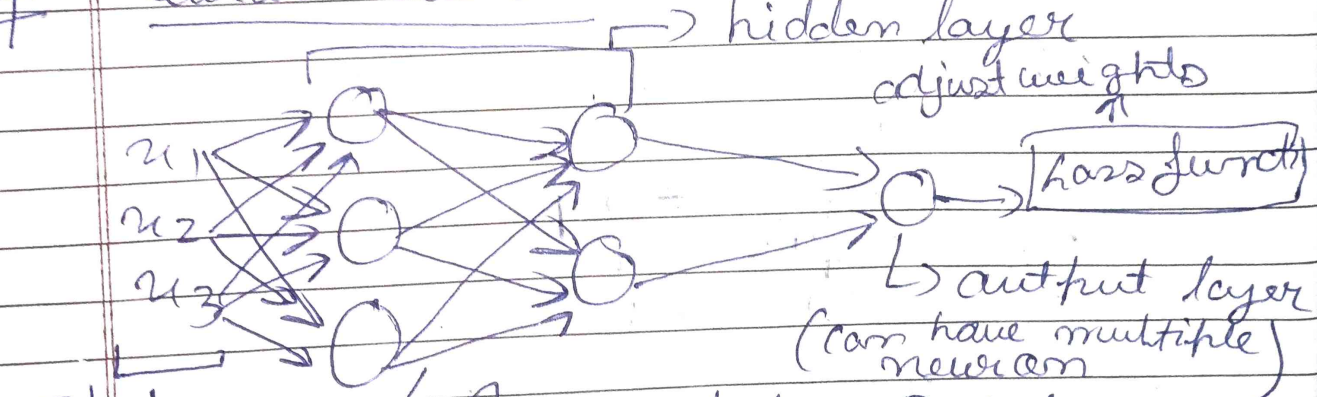
$Z = W^T X + B$ → bias vector
↓ ↳ input
↳ all weights in some vector form
all pre activation of layer

$\sigma(Z)$ = activation of whole layer

$$= \begin{bmatrix} \sigma(z_1) \\ \sigma(z_2) \\ \sigma(z_3) \end{bmatrix}$$

→ Different layer can have different activation function

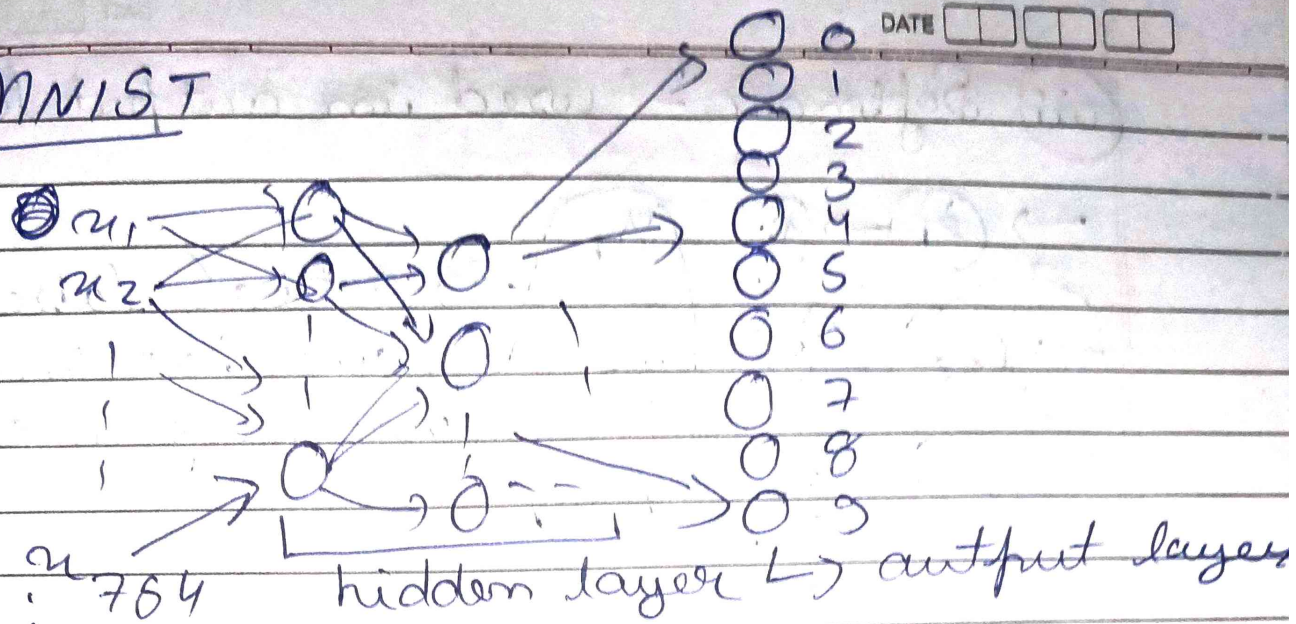
Neural Network :



→ each layer can have different activation function
→ output can be many or one

DATE

MNIST



↳ output layer

↳ grayScale value of MNIST image

→ output layer uses Softmax & output probability

→ neuron with max probability is chosen as prediction

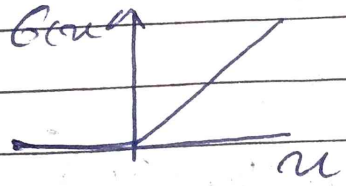
Activation function:

→ Takes z and give output $\sigma(z)$

(i) ReLU

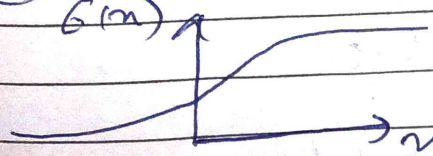
$\sigma(x) = \max(0, x)$

$\text{Derivative} = \sigma'(x) = \begin{cases} 1 & x > 0 \\ 0 & x \leq 0 \end{cases}$



(ii) Sigmoid

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

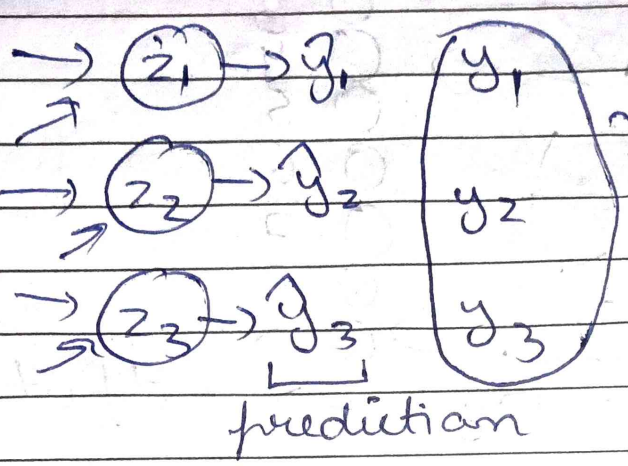


$\text{Derivative} = \sigma(x)(1-\sigma(x))$

$\sigma'(x) = \sigma(x)(1-\sigma(x))$

PAGE

(iii) Softmax :- used in output layer



$$\hat{y}_1 = \frac{e^{z_1}}{e^{z_1}+e^{z_2}+e^{z_3}} \qquad \hat{y}_2 = \frac{e^{z_2}}{e^{z_1}+e^{z_2}+e^{z_3}}$$

$$\hat{y}_3 = \frac{e^{z_3}}{e^{z_1}+e^{z_2}+e^{z_3}}$$

$\to \hat{y}_1 + \hat{y}_2 + \hat{y}_3 = 1$

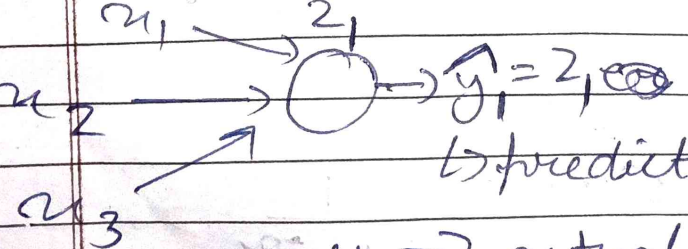
Derivative

$$\frac{\partial \hat{y}_i}{\partial z_j} = \begin{cases} -\hat{y}_i \hat{y}_j & k \neq j \\ \hat{y}_j(1-\hat{y}_j) & k = j \end{cases}$$

Calculating z_i require zs of all layer

Error function :-

(i) SSE (Sum of Square) :- in linear regression



↳ for a single sample

DATE

for multiple

$$L = \frac{L_1 + L_2 + \dots}{n}$$

$$= \frac{\sum_{n} (y_i - \hat{y}_i)^2}{n}$$

↳ optimal averaging

(iii) Cross Entropy loss :-

In classification (MNIST)

$$L = -\sum y_i \log \hat{y}_i \quad [\text{Single Sample}]$$

y_i = probability of ith class

eg $0 \to \hat{y}_1$ ⊕ In actual the class label is 2
$0 \to \hat{y}_2$ Hence $y = [0, 1, 0, 0, 0]$
$0 \to \hat{y}_3$ $y_1\ y_2\ y_3\ y_4\ y_5$
$0 \to \hat{y}_4$ ↳ one hot encoding
$0 \to \hat{y}_5$ for class 2
prediction

$$\text{Loss} = -[y_1 \log \hat{y}_1 + y_2 \log \hat{y}_2 + y_3 \log \hat{y}_3 + y_4 \log \hat{y}_4 + y_5 \log \hat{y}_5]$$

$$= -[0 + \log \hat{y}_2 + 0 + 0 + 0]$$

$$= -\log \hat{y}_2$$

↳ probability for class 2 given by model

PAGE

→ for multiple record

$$L = \frac{L_1 + L_2 \cdots}{n}$$

$$= \frac{\Sigma y \log y}{n} + \frac{\Sigma y \log y}{n} \quad \text{---}$$

Optimization :-

a function $z = f(x_1, x_2, x_3)$

decreases in direction of $\left\langle \frac{-\partial f}{\partial x_1}, \frac{-\partial f}{\partial x_2}, \frac{-\partial f}{x_3} \right\rangle$

I.e. - if we are at point (2, 5, 3)
and then we calculate $\left\langle \frac{-\partial f}{\partial x_1}, \frac{-\partial f}{\partial x_2}, \frac{-\partial f}{x_3} \right\rangle$
we can make a small nudge in that direction

$$\vec{R}_{next} = \vec{R}_{current} - \alpha \nabla f$$

As we make, ∇f also change
So we recalculate ∇f & keep making until minimal z is aquired

→ only gives local minima/maxima

Gradient descent :-

Loss function is dependent on weight & biases

$$L = f(w_1, w_2, w_3, b_1, b_2)$$

DATE

we calculate ∇f then move in negative direction

$$\nabla f = \left\langle \frac{\partial f}{\partial w_1}, \frac{\partial f}{\partial w_2}, \frac{\partial f}{\partial w_3}, \frac{\partial f}{\partial b_1}, \frac{\partial f}{\partial z} \right\rangle$$

Then update weight & bias

$$w_1' = w_1 - \alpha \frac{\partial f}{\partial w_1} \qquad w_3 = w_3 - \alpha \frac{\partial f}{\partial w_3}$$

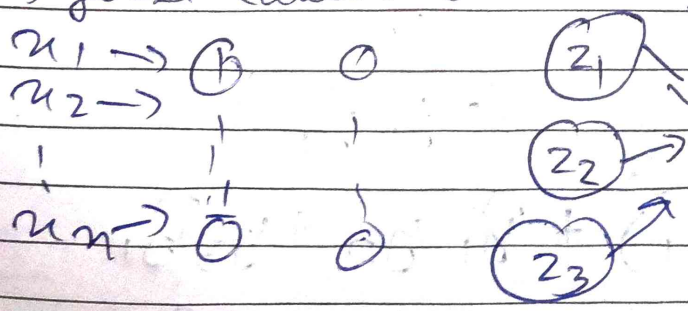
$$w_2' = w_2 - \alpha \frac{\partial f}{\partial w_2} \qquad b_1' = b_1 - \alpha \frac{\partial f}{\partial b_1}$$

$$b_2' = b_2 - \alpha \frac{\partial f}{\partial b_2}$$

Calculating $\frac{\partial L}{\partial w_i}$ & $\frac{\partial L}{\partial b_i}$:-

→ We use chains of derivative
→ for MNIST classification outer layer

For Single Sample
→ first calculate $\partial L / \partial z_i$ for each layer

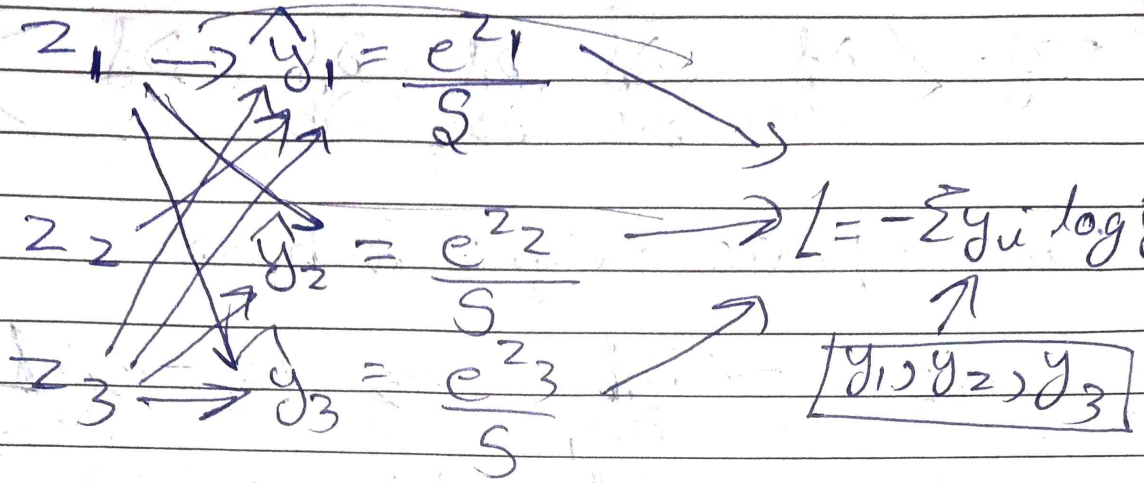


PAGE

DATE

Focus on outer layer first

Dependency graph:



→ $\hat{y}_i$ dependent on all z_i

$S = e^{z_1} + e^{z_2} + e^{z_3}$

$L = -\sum y_i \log \hat{y}_i = -[y_1 \log \hat{y}_1 + y_2 \log \hat{y}_2 + y_3 \log \hat{y}_3]$

$\Rightarrow \frac{\partial L}{\partial z_1}$ = all path from L to z_1

↳ z_1 = logit of first class

$$\Rightarrow \frac{\partial L}{\partial z_1} = \frac{\partial L}{\partial \hat{y}_1}\frac{\partial \hat{y}_1}{\partial z_1} + \frac{\partial L}{\partial \hat{y}_2}\frac{\partial \hat{y}_2}{\partial z_1} + \frac{\partial L}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial z_1}$$

$$= \left[\frac{-y_1}{\hat{y}_1}\hat{y}_1(1-\hat{y}_1) + \left(\frac{-y_2}{\hat{y}_2}\right)(-\hat{y}_2\hat{y}_1) + \left(\frac{-y_3}{\hat{y}_3}\right)(-\hat{y}_3\hat{y}_1)\right]$$

$$= -y_1(1-\hat{y}_1) + \hat{y}_1 y_2 + y_3\hat{y}_1$$

$$= -y_1 + y_1\hat{y}_1 + \hat{y}_1 y_2 + y_3\hat{y}_1$$

PAGE

DATE

$= -y_1 + \hat{y}_1(y_1 + y_2 + y_3)$

$= -y_1 + \hat{y}_1$ $\quad y_1 + y_2 + y_3 = 1$ [one hot]

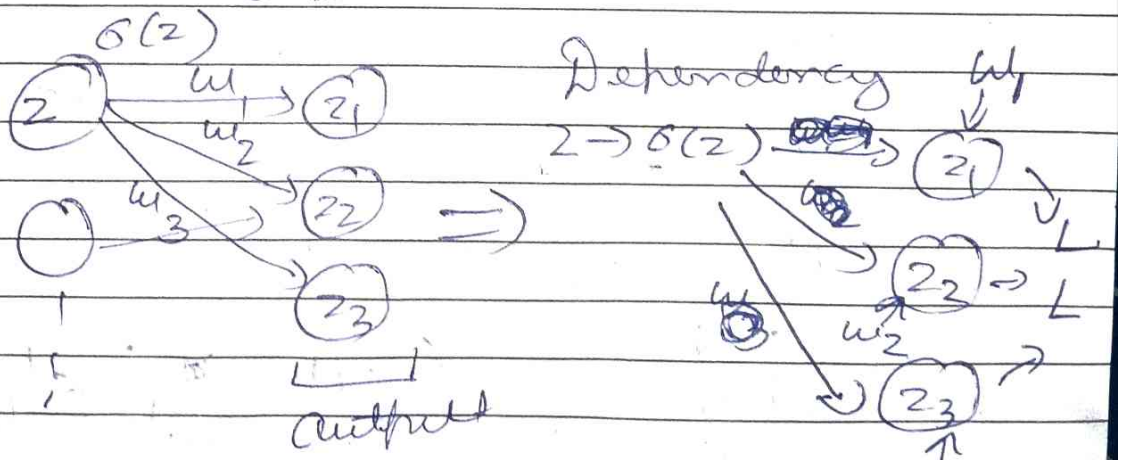
$= \hat{y}_1 - y_1$

$$\boxed{\frac{dL}{dz_i} = \hat{y}_i - y_i}$$ → In General

→ We calculate $\frac{\partial L}{\partial z_1}, \frac{\partial L}{\partial z_2}, \frac{\partial L}{\partial z_3}$ & so on

on for last layer

To calculate $\frac{\partial L}{\partial z_i}$ for previous layer



$$\frac{\partial L}{\partial z} = \frac{\partial L}{\partial z_1}\frac{\partial z_1}{\partial \sigma(z)}\frac{\partial \sigma(z)}{\partial z} + \frac{\partial L}{\partial z_2}\frac{\partial z_2}{\partial \sigma(z)}\frac{\partial \sigma(z)}{\partial z} + \frac{\partial L}{\partial z_3}\frac{\partial z_3}{\partial \sigma(z)}\frac{\partial \sigma(z)}{\partial z}$$

$$= \frac{\partial L}{\partial z_1} \cdot w_1 \cdot \sigma'(z) + \frac{\partial L}{\partial z_2} \cdot w_2 \cdot \sigma'(z) + \frac{\partial L}{\partial z_3} w_3 \cdot \sigma'(z)$$

PAGE

DATE

Hence

$$\frac{\partial L}{\partial z} = \left(\sum \frac{\partial L}{\partial z_i} \cdot w_i \right) \sigma'(z)$$

↳ derivative of activation unit z

In matrix form

$$\frac{\partial L}{\partial z} = (\delta^T w_e)\, \sigma'(z)$$

↳ weight vector of connected edge

$$\delta = \begin{bmatrix} \partial L/\partial z_1 \\ \partial L/\partial z_2 \\ \partial L/\partial z_3 \end{bmatrix} = \text{delta vector}$$

For whole layer

↳ To find $\partial L/\partial z$

z_{11} z_{21}

z_{12} z_{22}

z_{13} z_{23}

① ② ↳ $\frac{\partial L}{\partial z}$ known

$$z = \begin{bmatrix} z_1 \\ z_{12} \\ z_{13} \end{bmatrix}$$

$$\delta = \begin{bmatrix} \partial L/\partial z_{11} \\ \partial L/\partial z_{12} \\ \partial L/\partial z_{13} \end{bmatrix} = (w \; \delta) \sigma'(z)$$

$i \times n$ $n \times 1$

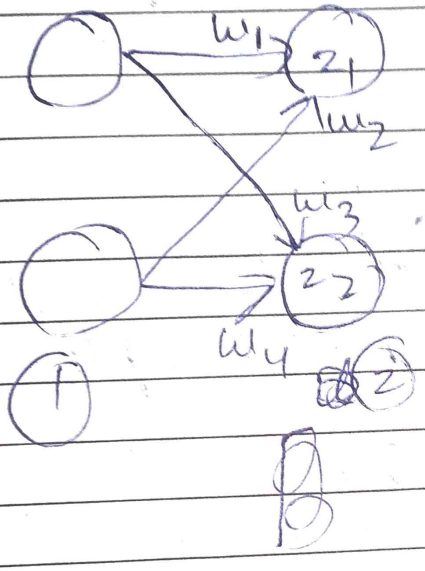
w = weight of layer (2)

δ = delta of layer 2

$\sigma'(z)$ = activation derivative of layer 1

PAGE

eg



W_2 = $\begin{bmatrix} w_1 & w_3 \\ w_2 & w_4 \end{bmatrix}$

↳ weight matrix of layer 2

$W_2 \delta = \begin{bmatrix} w_1 & w_3 \\ w_2 & w_4 \end{bmatrix} \begin{bmatrix} \partial L/\partial z_1 \\ \partial L/\partial z_2 \end{bmatrix}$

$= \begin{bmatrix} (\partial L/\partial z_1) w_1 + (\partial L/\partial z_2) w_3 \\ (\partial L/\partial z_1) w_2 + (\partial L/\partial z_2) w_4 \end{bmatrix}$

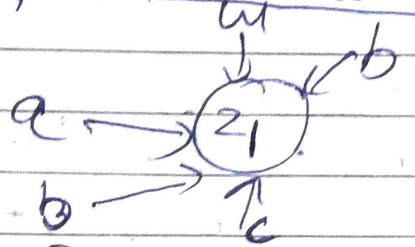
↳ matches with chain of derivative

→ May need to use $w^T \delta$ if weight of neuran are row vector.

DATE

→ Now all other layer $\partial\delta$ can be calculated with only using next layer δ without exploring full path derivative

To calculate $\partial L/\partial w$:-



a = activation from previous layer or input

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial z_1}\frac{\partial z_1}{\partial w} = \left(\frac{\partial L}{\partial z_1}\right)\cdot a$$

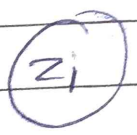
Activation of weight w

$$z_1 = wa + w_1 b + w_2 c + b$$

$$\frac{\partial z_1}{\partial w} = a$$

$$\boxed{\frac{\partial L}{\partial w_i} = \left(\frac{\partial L}{\partial z_i}\right) a} \rightarrow \text{for Single Sample}$$

To calculate $\partial L/\partial b$:-



$$\frac{\partial b}{\partial L} = \frac{\partial L}{\partial z_1}\frac{\partial z_1}{\partial b} = \frac{\partial L}{\partial z_1}$$

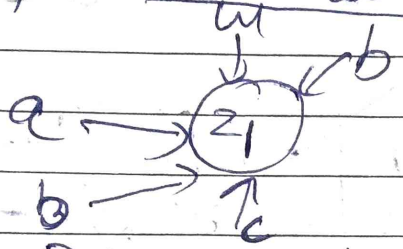
$$\frac{\partial z_1}{\partial b} = 1$$

PAGE

DATE

→ Now all other layer $\partial\delta$ can be calculated with only using next layer δ without exploring full path derivative

To calculate $\partial L/\partial w$:-



a = activation from previous layer or input

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial z_1}\frac{\partial z_1}{\partial w} = \left(\frac{\partial L}{\partial z_1}\right)\cdot a$$

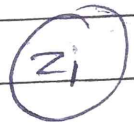
(activation of weight w)

$$z_1 = wa + w_1 b + w_2 c + b$$

$$\frac{\partial z_1}{\partial w} = a$$

$$\boxed{\frac{\partial L}{\partial w_i} = \left(\frac{\partial L}{\partial z_i}\right) a}$$ → for single sample

To calculate $\partial L/\partial b$:-



$$\frac{\partial b}{\partial L} = \frac{\partial L}{\partial z_1}\frac{\partial z_1}{\partial b} = \frac{\partial L}{\partial z_1}$$

$$\frac{\partial z_1}{\partial b} = 1$$

PAGE

DATE

$$\frac{\partial L}{\partial b_i} = \frac{\partial L}{\partial z_i}$$

Now after calculating $\partial L/\partial w_i$, $\partial L/\partial b_i$ we update weight and bias

$$w_i' = w_i - \alpha \frac{\partial L}{\partial w_i} \qquad b_i' = b_i - \alpha \frac{\partial L}{\partial b_i}$$

In matrix form

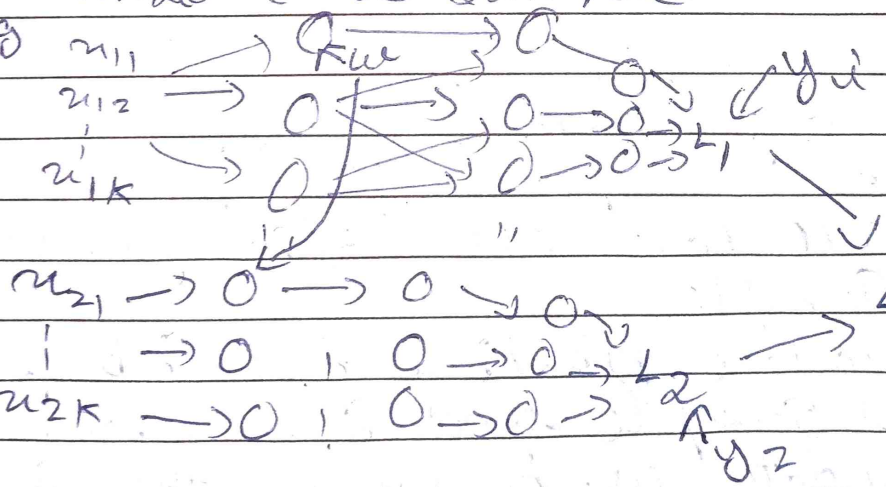
$$W = W - \alpha \nabla W \qquad B = B - \alpha \nabla B$$

Batch gradient descent :-

In Batch GD loss of whole batch is considered

Consider 2 sample

eg



Consider the weight of w of a neuron.

DATE

In dependency graph the weight (w) affect both L_1 & L_2
(pre activation changes per Sample hence $\partial L/\partial z_i$ changes)

→ weight has path through all Sample
Hence

$$w = w - \alpha\left(\sum \frac{\partial L}{\partial w}\right) \quad \text{(b)}$$

↳ Sample Sum of gradient

Similarly for b $\quad b = b - \alpha\left(\sum \frac{\partial L}{\partial b}\right)$

However $\frac{\partial L}{\partial z}$ are not connected to other graph & (b) z is different for different Sample hence $\frac{\partial L}{\partial z}$ is not added

Batch algorithm :-
for e in epochs (100):
calculate all $\partial L/\partial z$ for all Sample.
Calculate (b) $\partial L/\partial w$ & $\partial L/\partial b$ for all Sample and Sum them
Update w, b using the Sum

DATE

Stochastic
for epoch (100):
    for each Sample in S:
        calc all $\partial L/\partial z$, $\partial L/\partial w$, $\partial L/\partial b$ for S
        update w, b

Adam Optimizer & Momentum based learning rates :