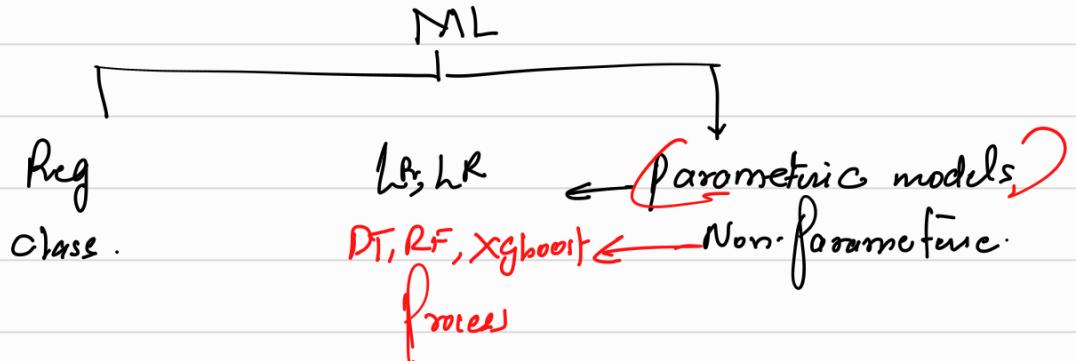
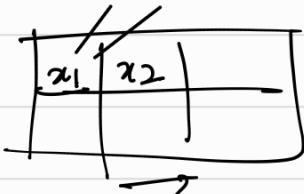


Deep learning

$$y = f(x) + \epsilon$$



$$y \sim f(x)$$



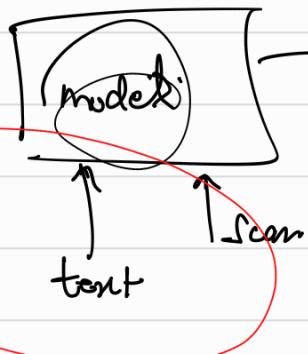
①
architecture

$x \rightarrow$ (video, audio, text, tabular data, graph).

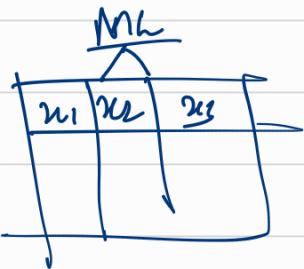
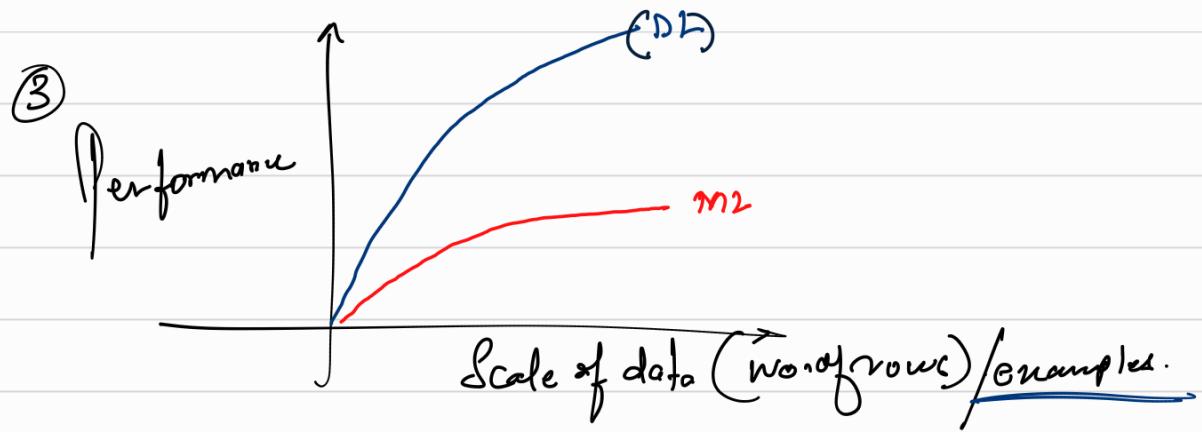
multimodality

②

Tabular data



Diagnosis

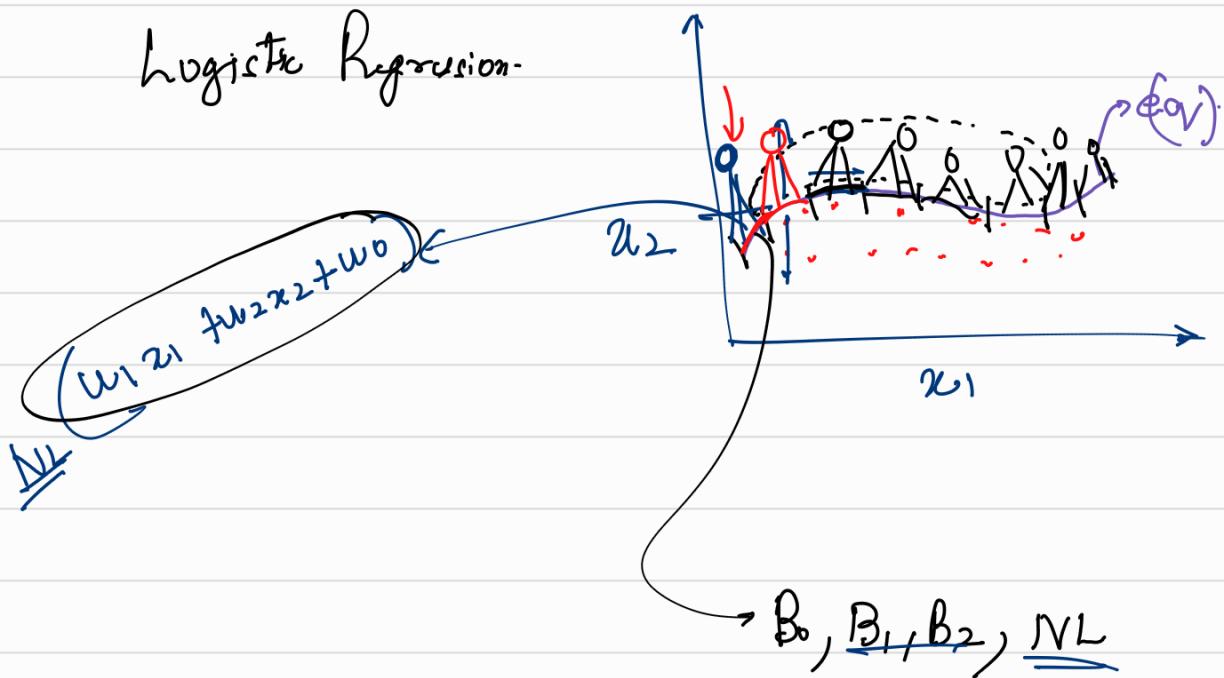


$D_h \rightarrow$ automated feature engineering

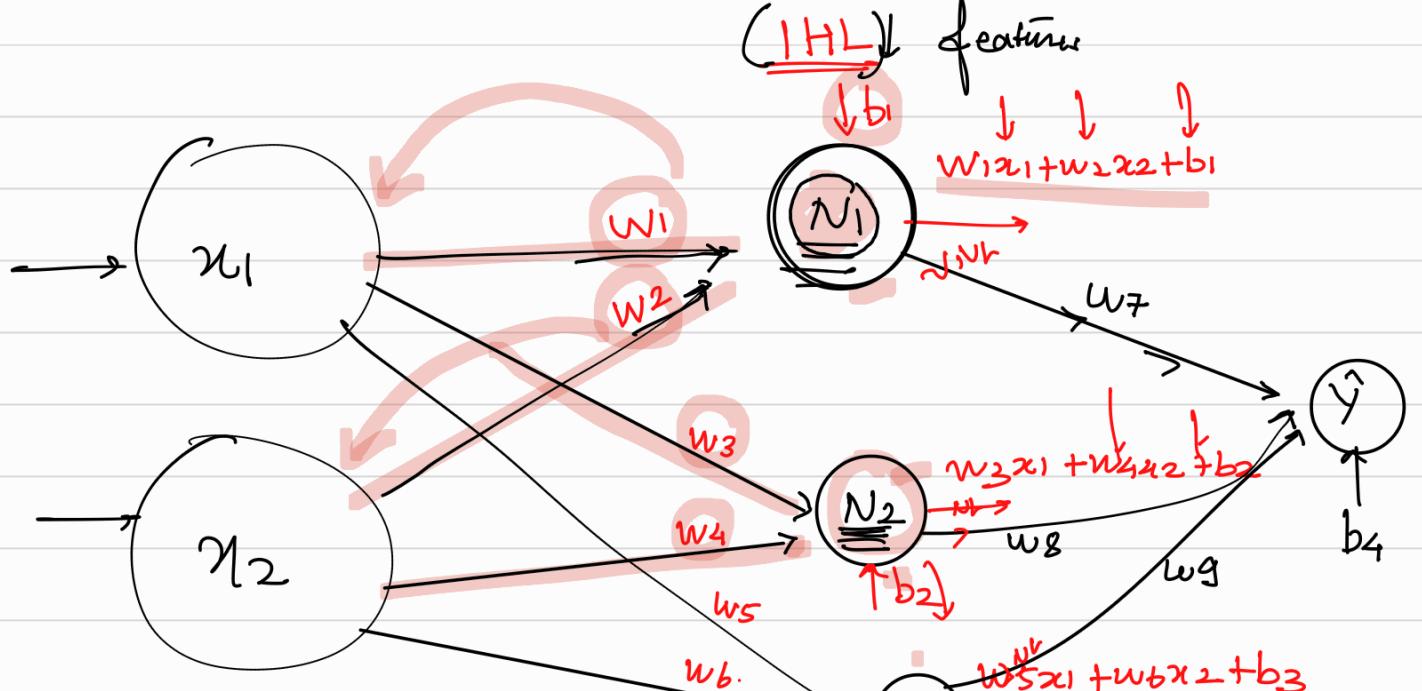


What is a Neuron

Logistic Regression-



m_1	x_1	y



$$N_1 = \underbrace{(w_1 x_1 + w_2 x_2 + b_1)}_{\text{feature}}, \quad N_2 = \underbrace{(w_3 x_1 + w_4 x_2 + b_2)}_{\text{feature}}, \quad N_3 = \underbrace{(w_5 x_1 + w_6 x_2 + b_3)}_{\text{feature}}$$

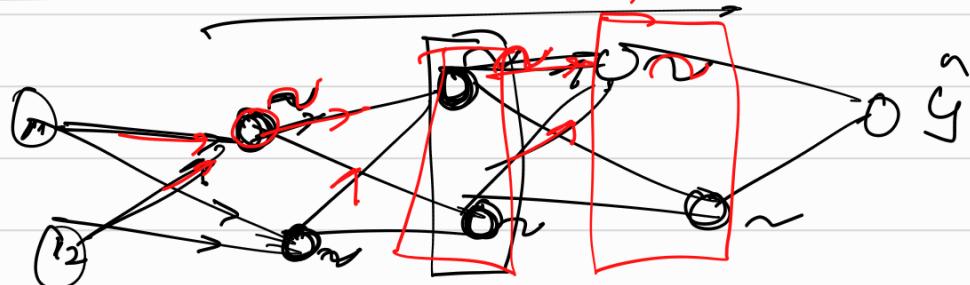
$$\hat{y} = w_7 N_1 + w_8 N_2 + w_9 N_3 + b_4$$

$$\Rightarrow \hat{y} = \underbrace{w_7}_{\text{feature}} \underbrace{(w_1 x_1 + w_2 x_2 + b_1)}_{\text{feature}} + \underbrace{w_8}_{\text{feature}} \underbrace{(w_3 x_1 + w_4 x_2 + b_2)}_{\text{feature}} + \underbrace{w_9}_{\text{feature}} \underbrace{(w_5 x_1 + w_6 x_2 + b_3)}_{\text{feature}} + b_4$$

$$\Rightarrow \hat{y} = x_1 (w_7 w_1 + w_8 w_3 + w_9 w_5) + x_2 (w_7 w_2 + w_8 w_4 + w_9 w_6)$$

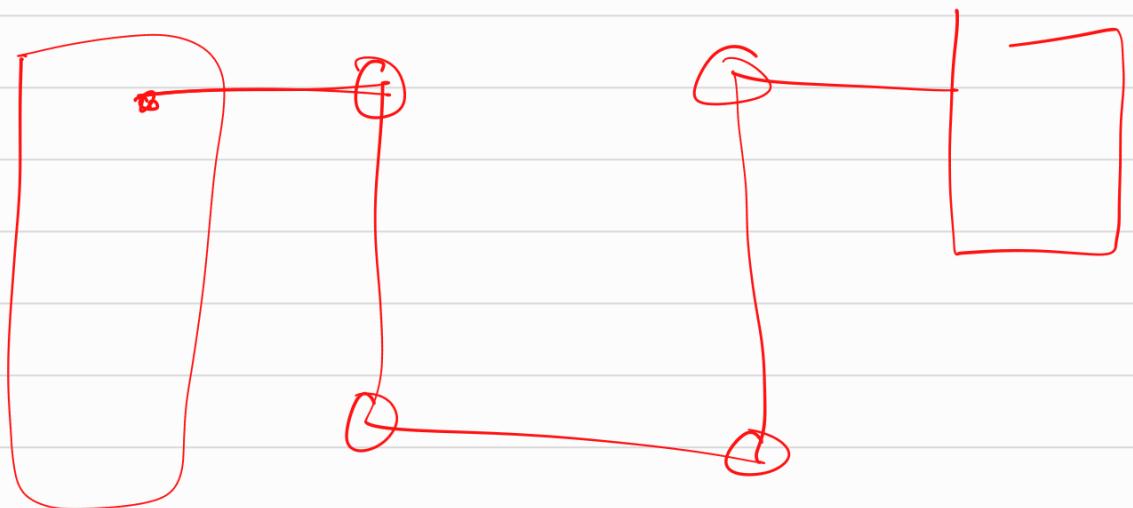
$$+ w_7 b_1 + w_8 b_2 + w_9 b_3 + b_4$$

$$\Rightarrow \hat{y} = A x_1 + B x_2 + C$$



1 H_L

→ nth H_L

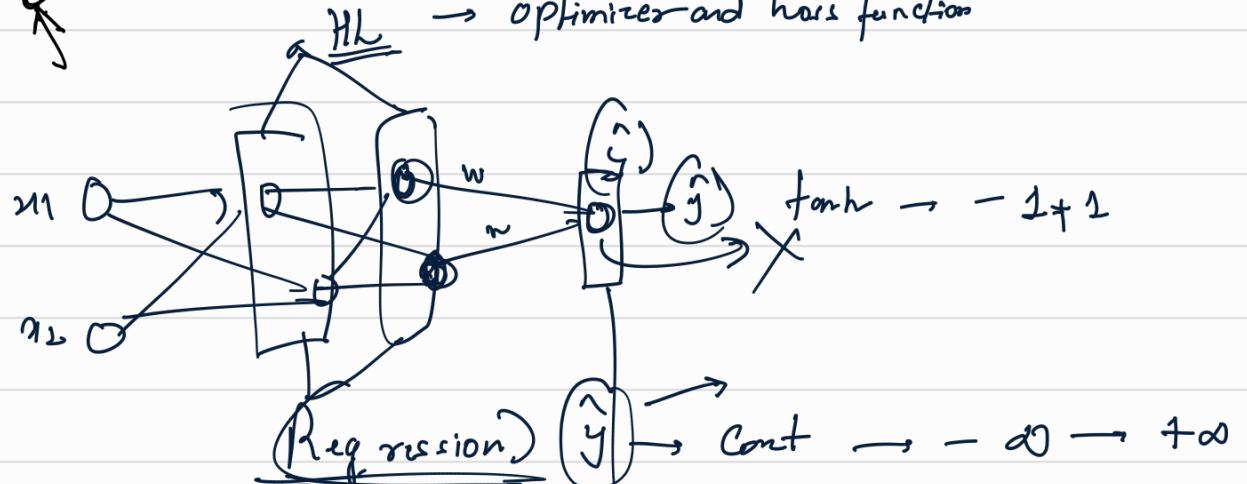
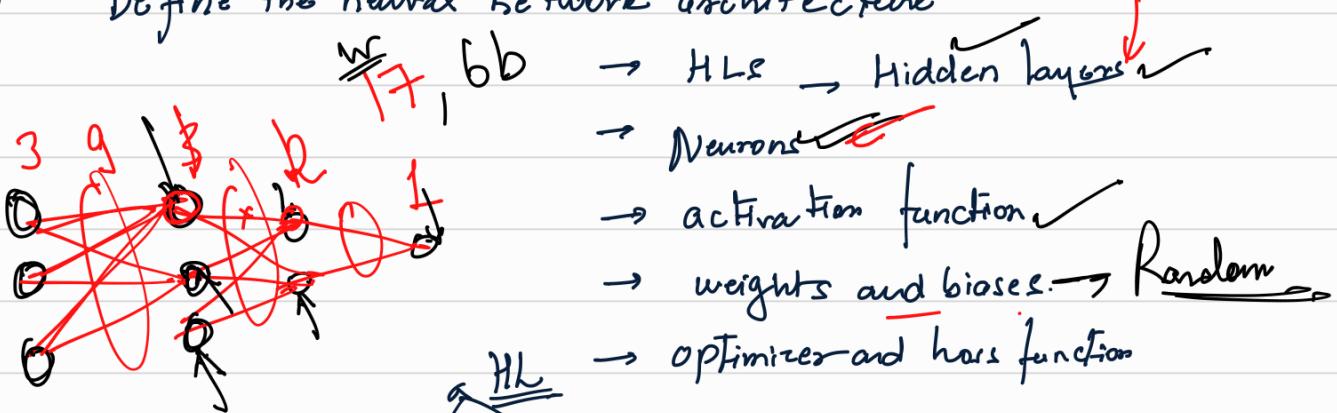


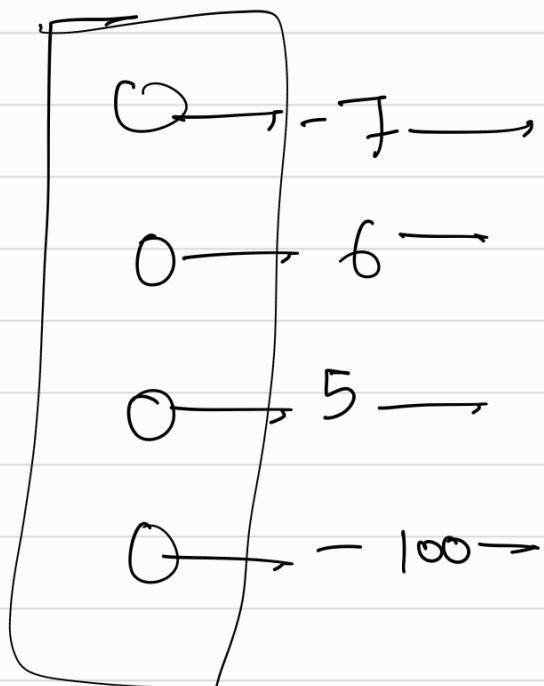
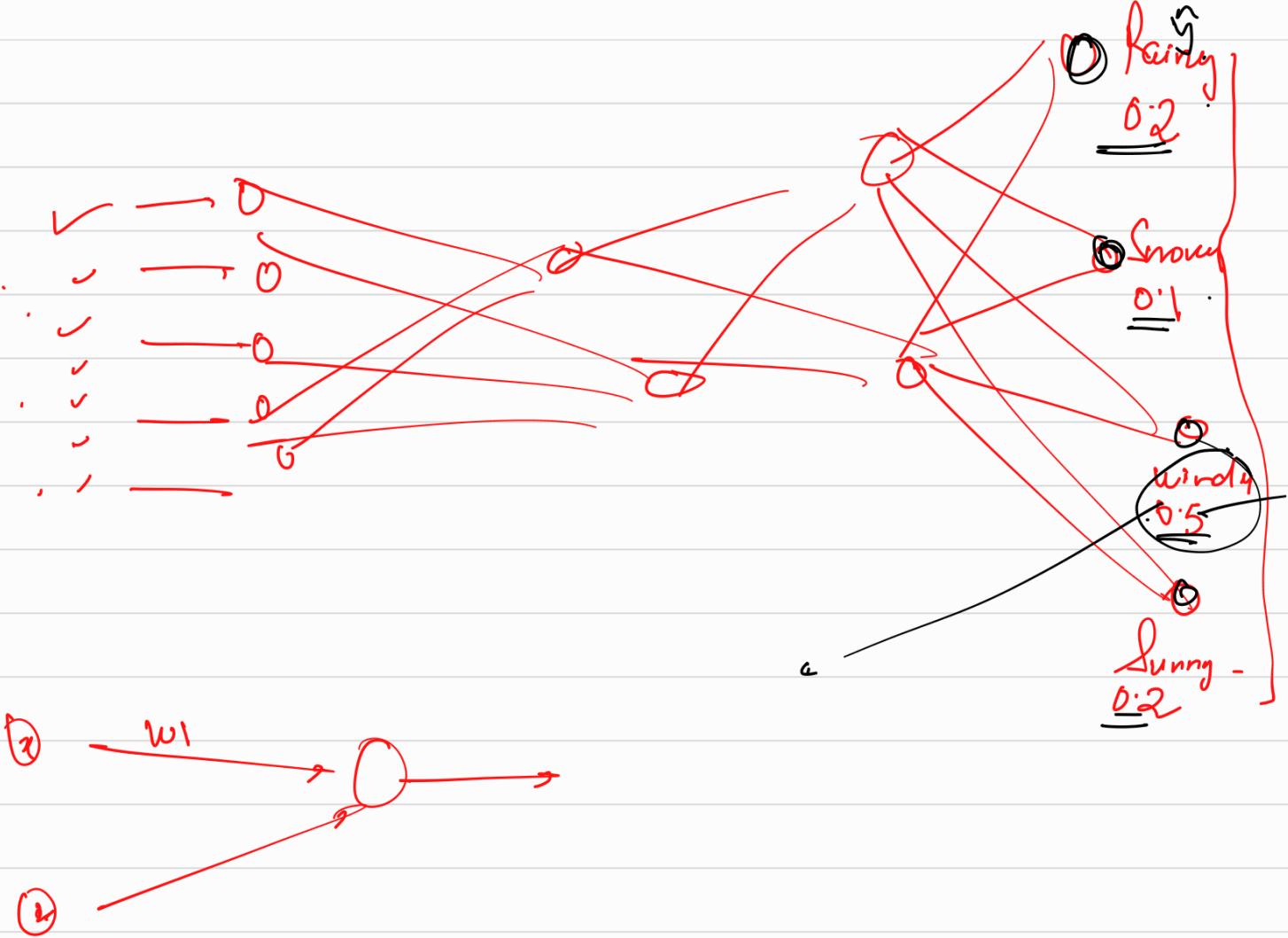
Gradient Descent

: minimise the loss by updating the W, B .
for all neurons.

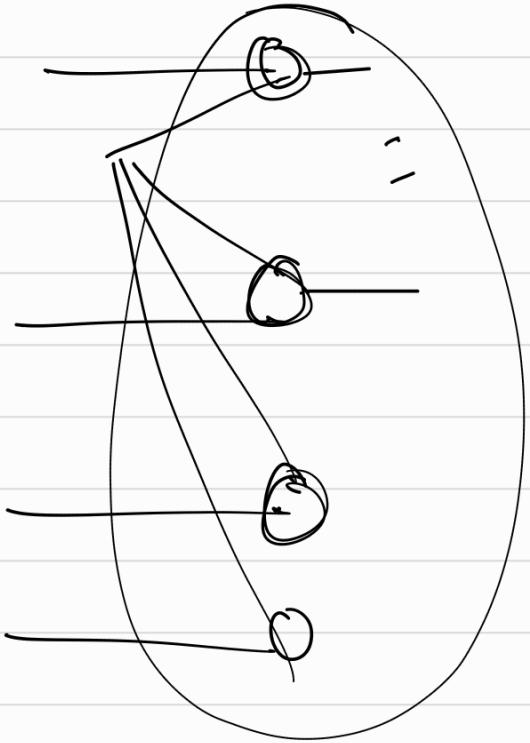
RanCham

1) Define the neural network architecture





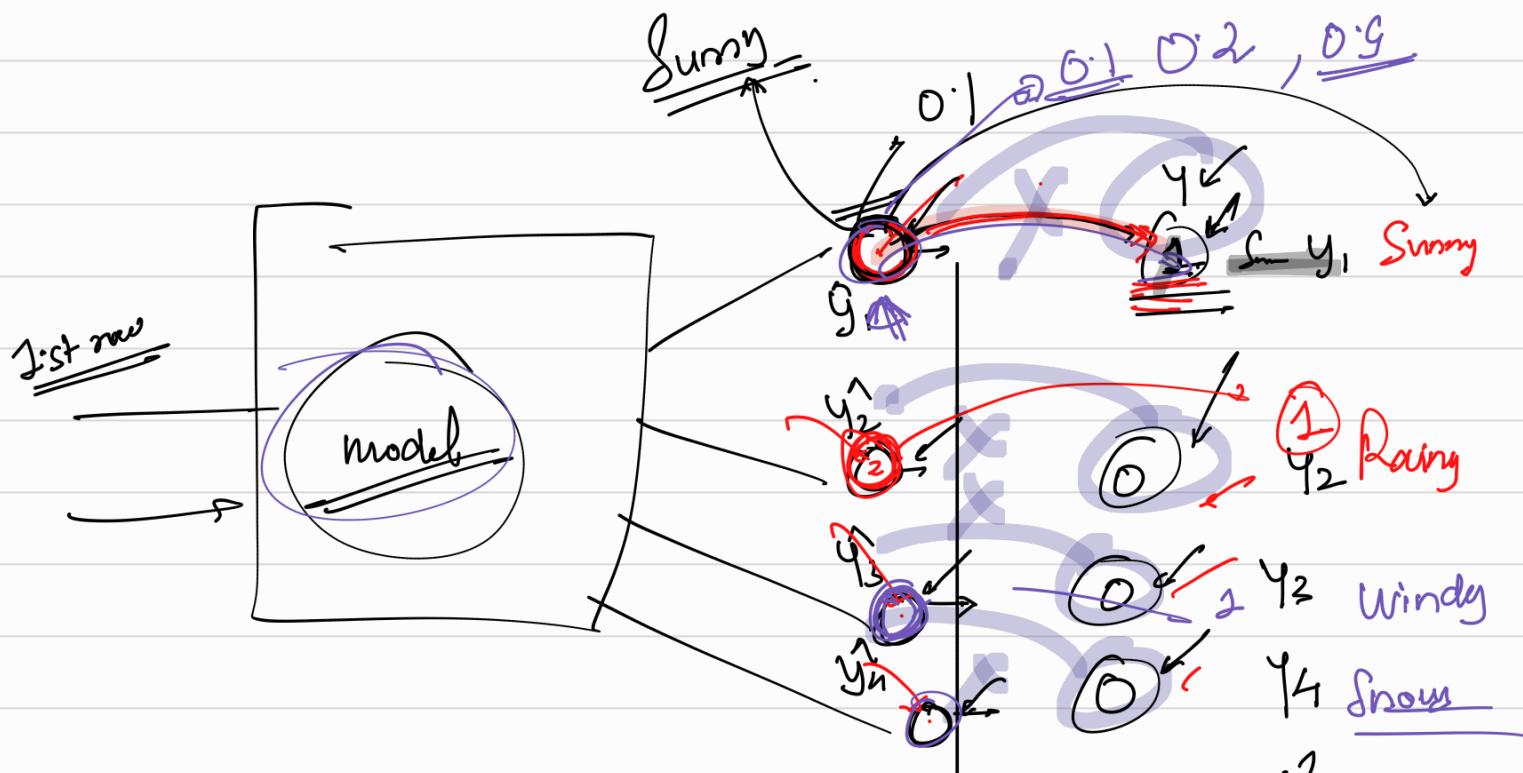
x_1	x_2	Weather (y)	$y-Su$	$y-Wi$	$y-Ra$	w_i	y
		Sunny	1	0	0	0	0
		Rainy	0	1	0	0	0
		Windy	0	0	1	1	0
		Sunny	1	0	0	0	0
		Shady	0	0	0	0	1



0

0

0
0



$$\begin{aligned}
 \text{log hrs} &= \text{for all hours of data given} - \left(\hat{y}_1 \log(\underline{\hat{y}_1}) + \hat{y}_2 \log \underline{\hat{y}_2} \right. \\
 &\quad \left. + \hat{y}_3 \log \underline{\hat{y}_3} + \hat{y}_4 \log \underline{\hat{y}_4} \right) \\
 &\quad - (\log 0.2) \\
 &= \underline{0.69}
 \end{aligned}$$

$$\rightarrow \left(\log 0.8 \right) = 0.09$$

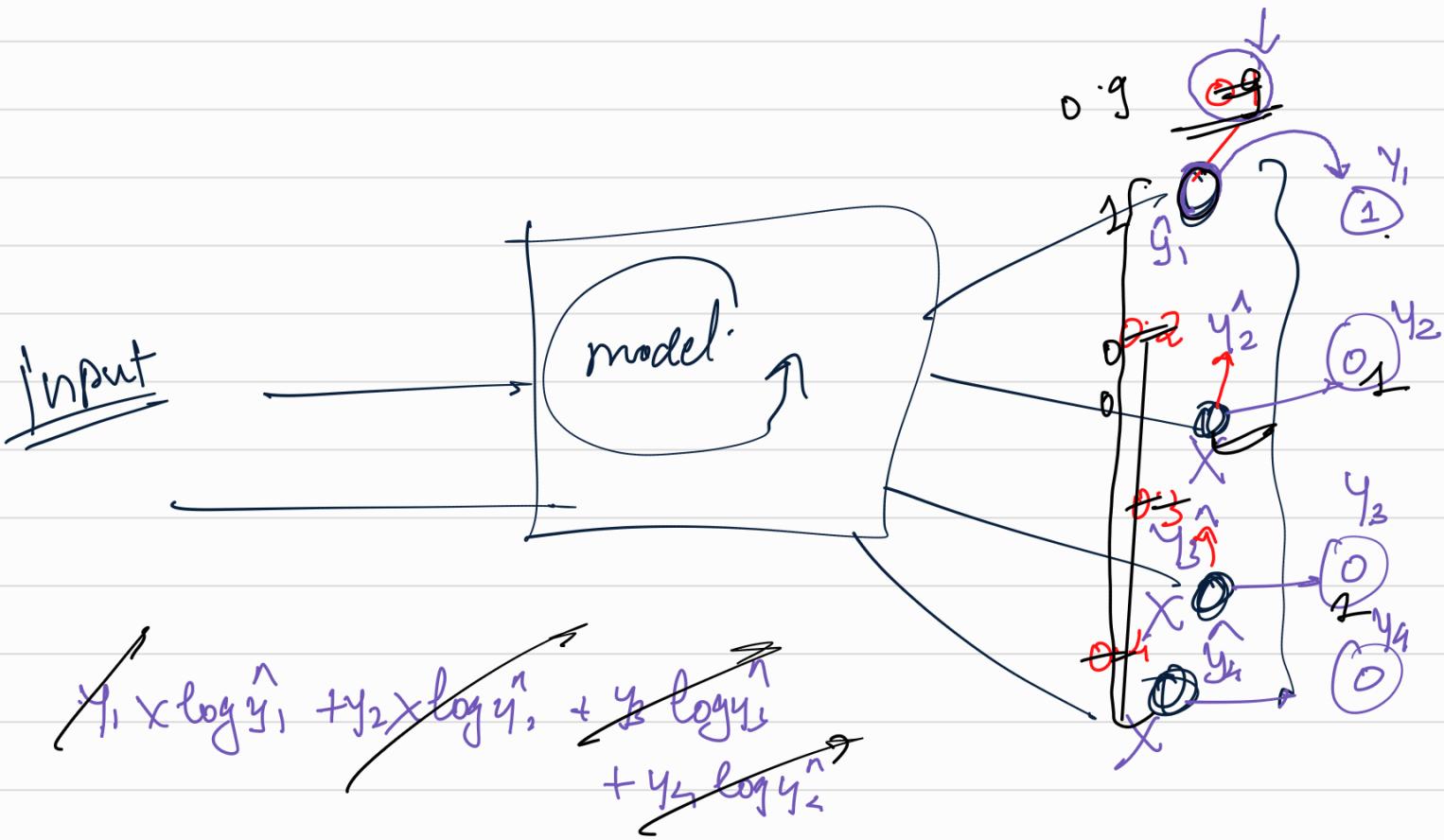
$$-\log(0.95) = 0.02$$

$$\log_1 = 0$$

$$y_1 \log \hat{y}_1 + y_2 \log \hat{y}_2 + y_3 \log \hat{y}_3 + y_4 \log \hat{y}_4$$

~~$y_1 \log y_1 + y_2 \log y_2 + y_3 \log y_3 + y_4 \log y_4$~~

$$\log(0.1) = -1$$



$$-\left(\log \hat{y}_i\right)$$

$$\cancel{\log 1 = 0}$$

$$-\left(\log 0.1\right)$$

↓

$$-\left(0\right)$$

VS

VS

$$-\cancel{\log 0.9}$$

↓

$$0.045$$

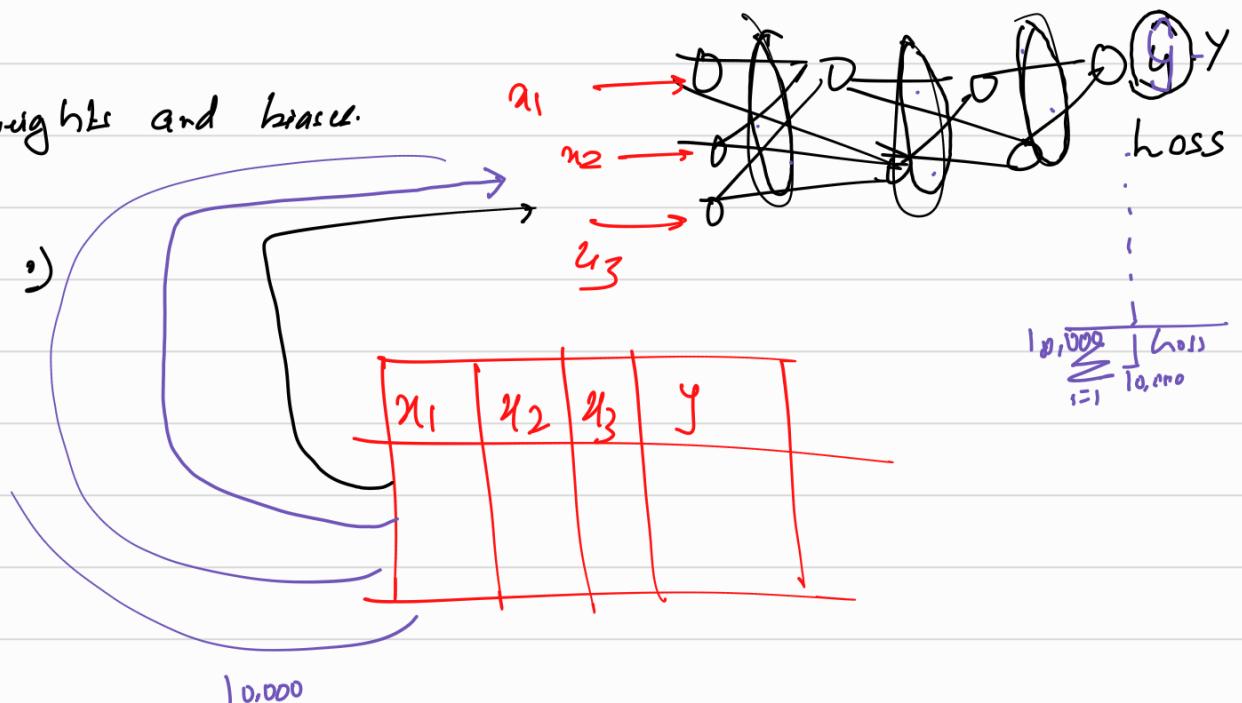
0
0
0

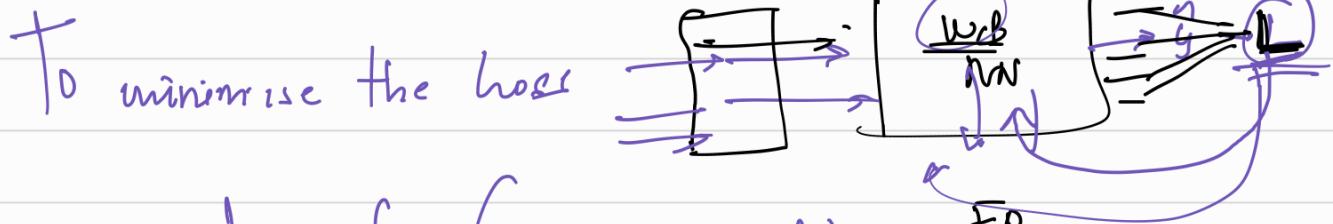
$$(0)$$

softmax, ohe

log loss

Random weights and biases.

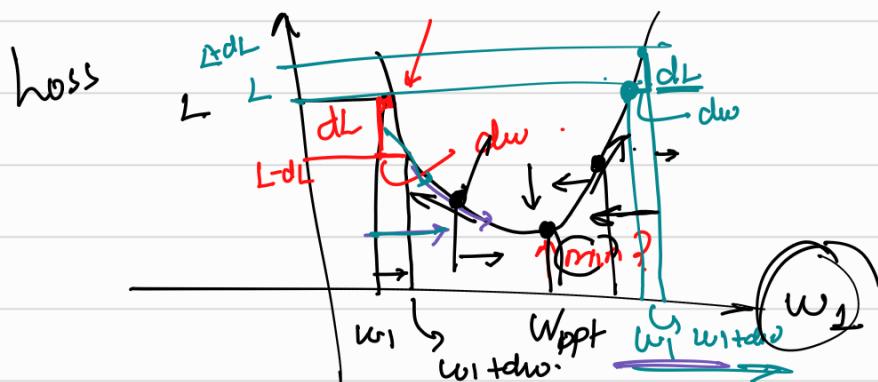
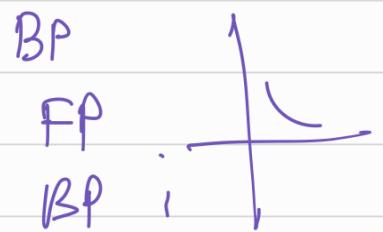




$$L = f(y_{\text{predicted}}, y_{\text{actual}})$$

$$L = f(\underline{w}, \underline{b}, \underline{z}), \underline{y_{\text{actual}}})$$

$$L = f(w, b)$$



$$\frac{dw}{dL} = -\text{ve.}$$

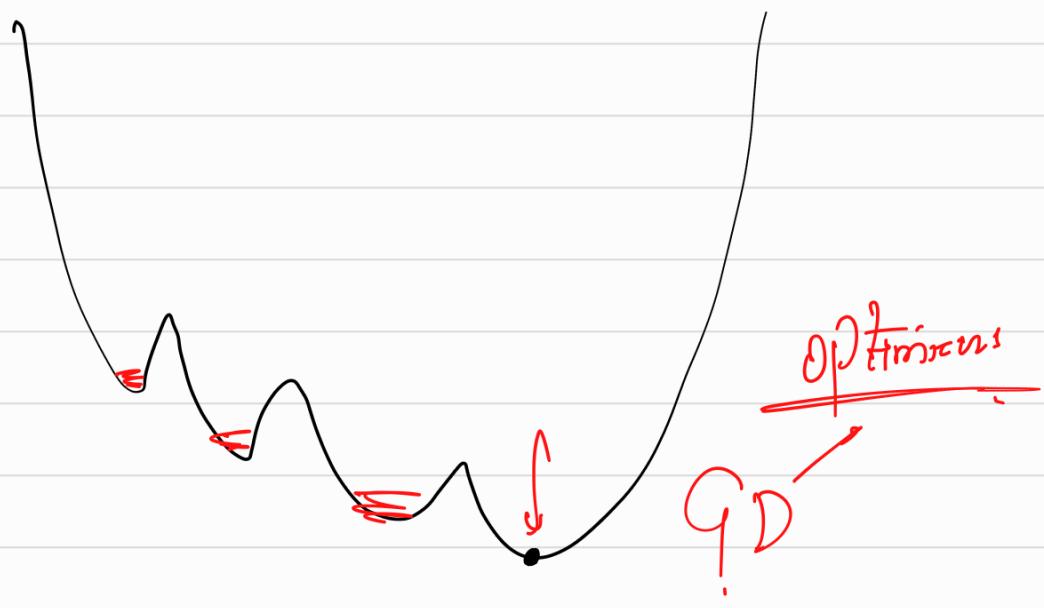
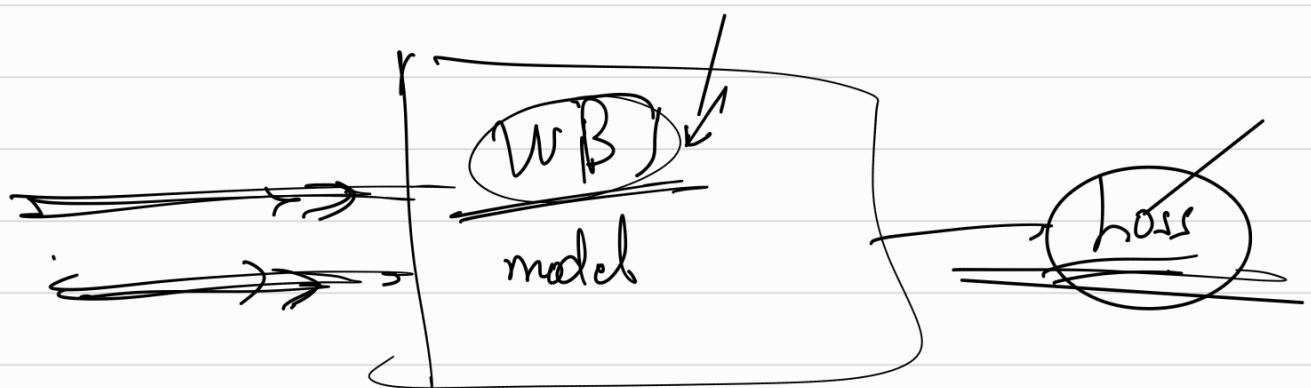
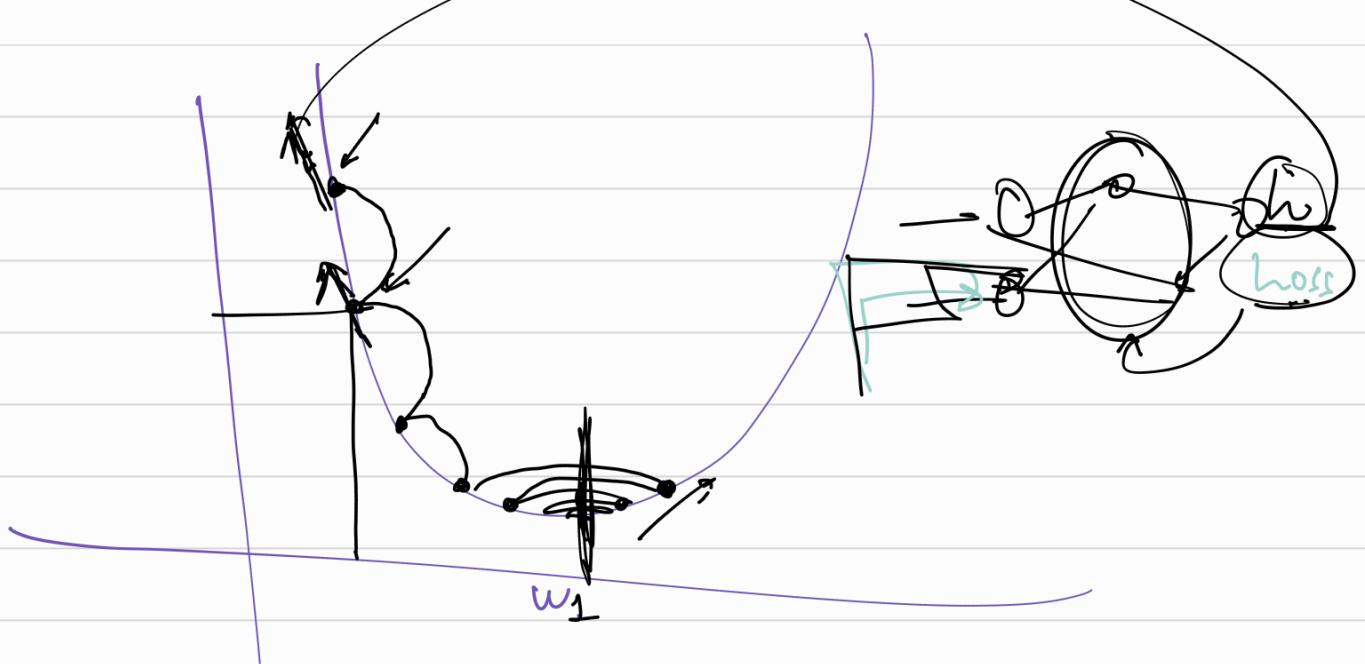
which side works out +ve
more

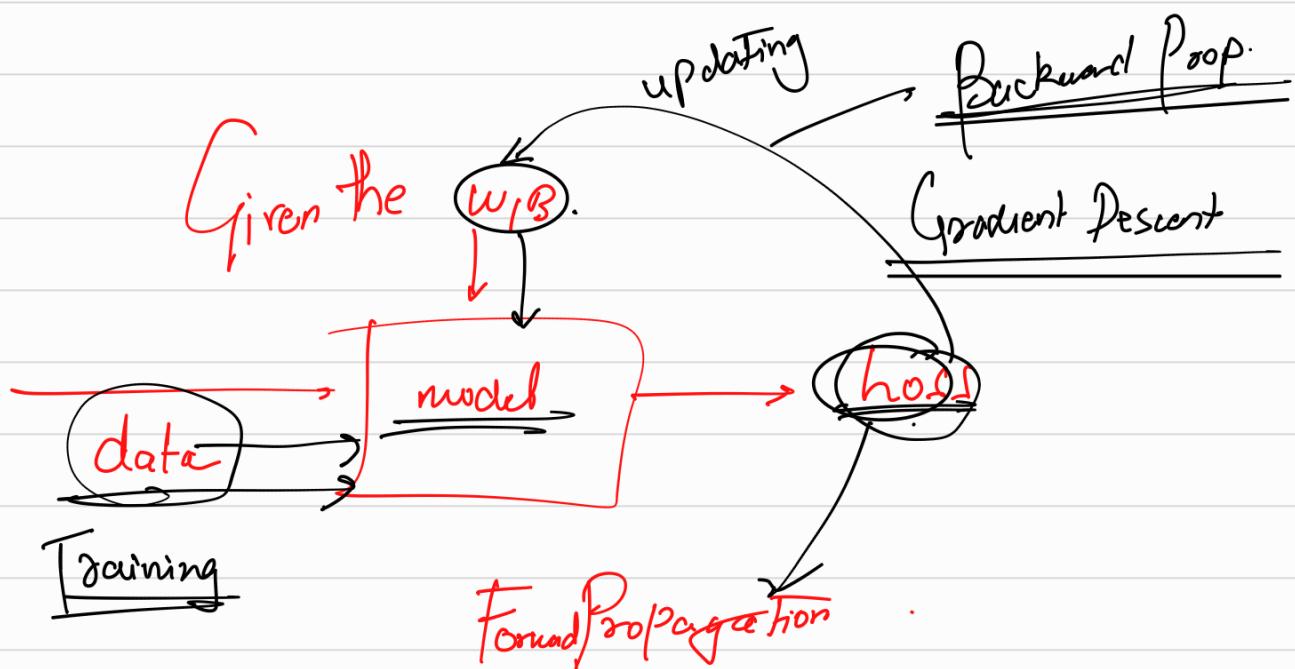
$$\frac{dw}{dL} = +\text{ve}$$

$$(w = -\text{ve.})$$

$$w_i(\text{new}) = w_i(\text{old}) - \alpha \left(\frac{dw}{dL} \right)$$

Applicable for all weights and biases.





$(\text{IFP} + \text{IBP}) \rightarrow \underline{\text{epoch}}$

$(\text{IFP}, \text{IBP}) \rightarrow \text{epoch}$ → $\downarrow \text{epoch}$

$\boxed{\text{IFP} + \text{IBP}}$

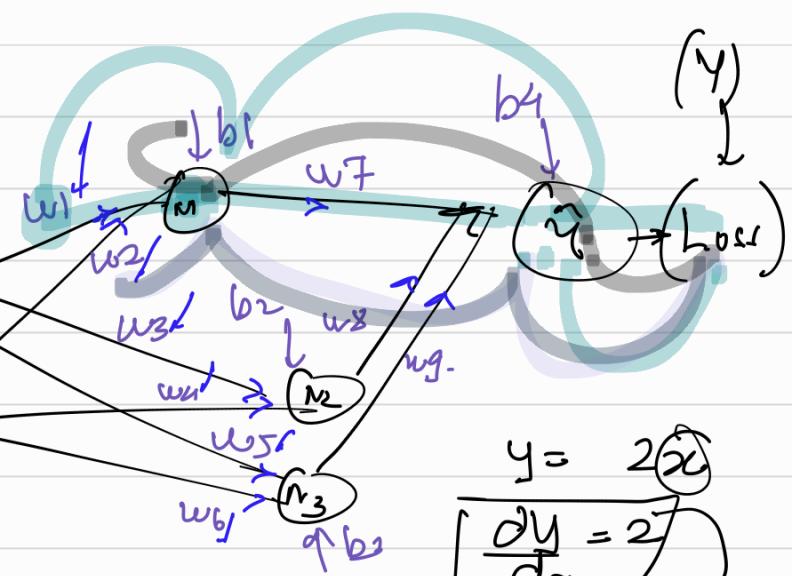
Lecture-3

$$N_1 = w_1 x_1 + w_2 x_2 + b_1 \quad (1)$$

$$N_2 = w_3 x_1 + w_4 x_2 + b_2$$

$$N_3 = w_5 x_1 + w_6 x_2 + b_3$$

$$\hat{y} = w_7 N_1 + w_8 N_2 + w_9 N_3 + b_4$$



$$y = 2x$$

$$\frac{\partial y}{\partial x} = 2$$

$$\frac{\partial L}{\partial \hat{y}} = \sum_{i=1}^n \frac{1}{n} (y_i - \hat{y}_i)^2 \rightarrow \text{MSE}$$

$$\frac{\partial L}{\partial w_7} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial w_7} \Rightarrow \sum_{i=1}^n \frac{1}{n} \times 2 \times \frac{(y_i - \hat{y}_i)}{\partial \hat{y}} \times \underset{x-1}{\cancel{N_1}}$$

$$\frac{\partial h}{\partial w_9}, \frac{\partial h}{\partial w_1}$$

6 mins

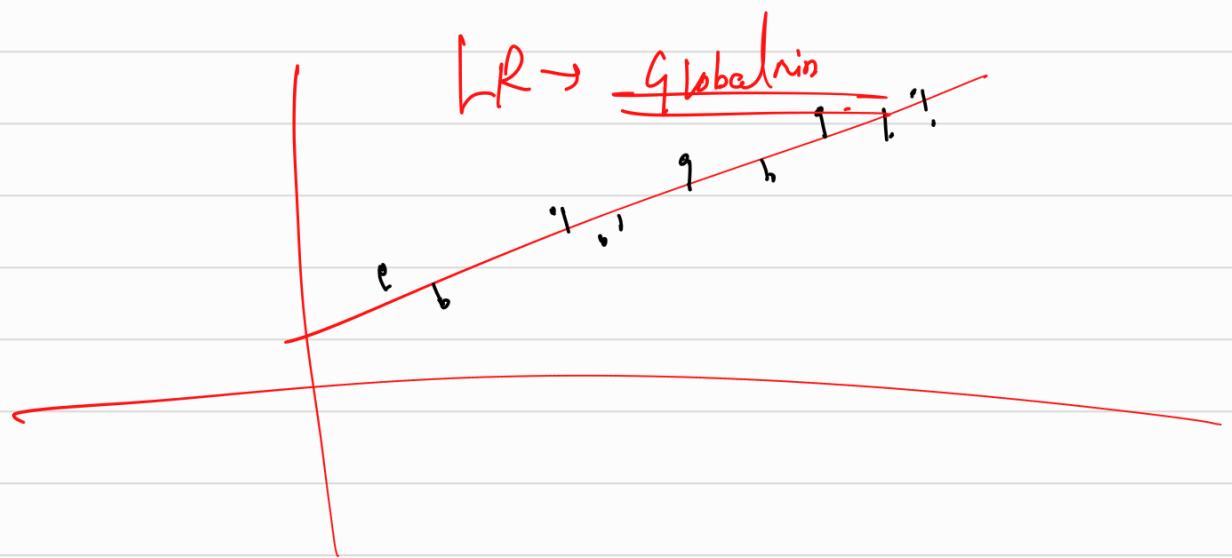
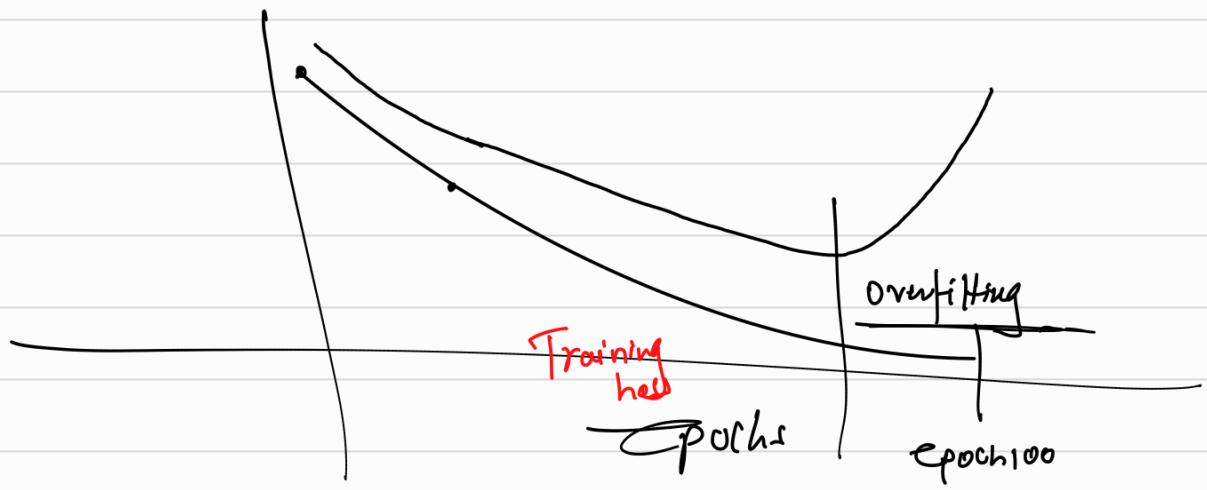
$$\frac{\partial L}{\partial w_9} = \left(\frac{\partial L}{\partial \hat{y}} \right) \times \frac{\partial \hat{y}}{\partial w_9} \rightarrow N_3$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial N_1} \times \frac{\partial N_1}{\partial w_1}$$

$$\hat{y} = w_7 N_1 + w_8 N_2 + w_9 N_3 + b_4$$

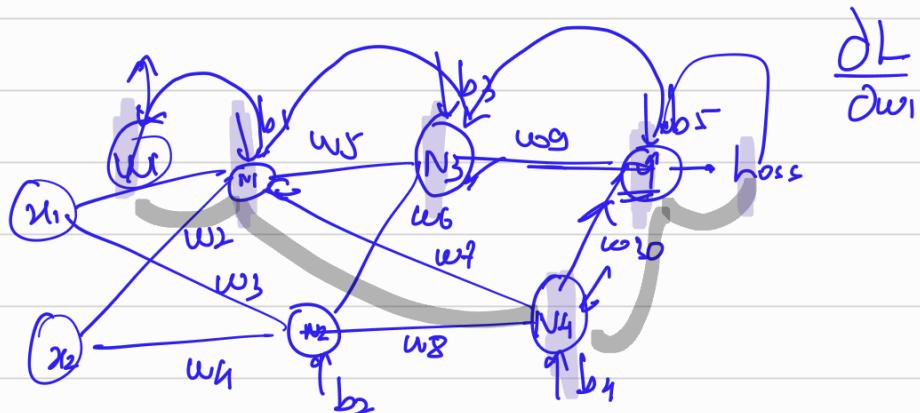
$$\frac{\partial \hat{y}}{\partial w_7} = \frac{\partial (\textcircled{w_7} \textcircled{N_1})}{\partial w_7} = N_1$$

$$\frac{\partial (x_1 x_2)}{\partial x} = 2$$



Activation functions

$$w_1x_1 + w_2x_2 + b_1$$

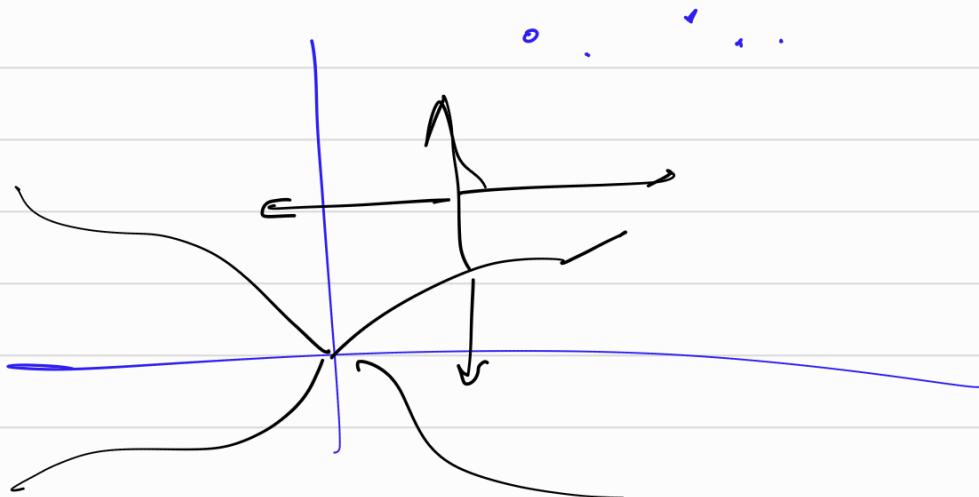


$\frac{\partial h}{\partial w_1} = \left(\frac{\partial L}{\partial y} \times \frac{\partial y}{\partial N_3} \times \frac{\partial N_3}{\partial N_1} \times \frac{\partial N_1}{\partial w_1} \right) + \left(\frac{\partial h}{\partial y} \times \frac{\partial y}{\partial N_4} \times \frac{\partial N_4}{\partial N_1} \times \frac{\partial N_1}{\partial w_1} \right)$.

$\frac{\partial L}{\partial w} = 1000000$
~~Terms = $N(N_L)^2$~~

$\frac{\partial h}{\partial w_1} = 0.00000^2$

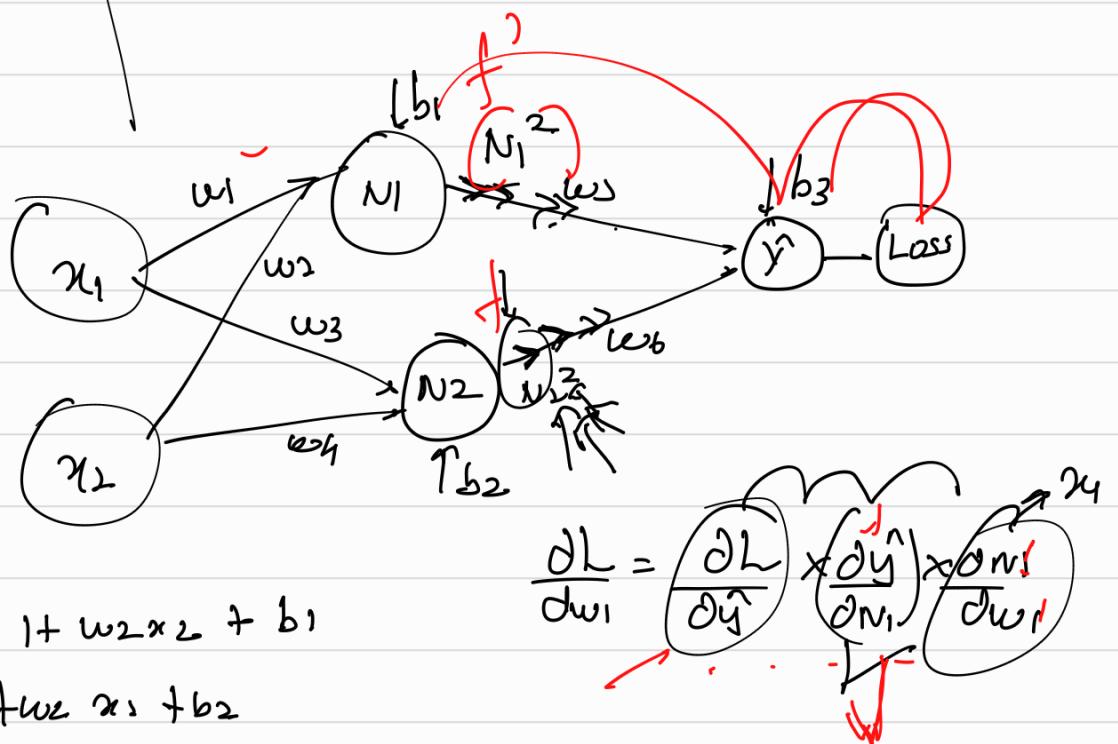
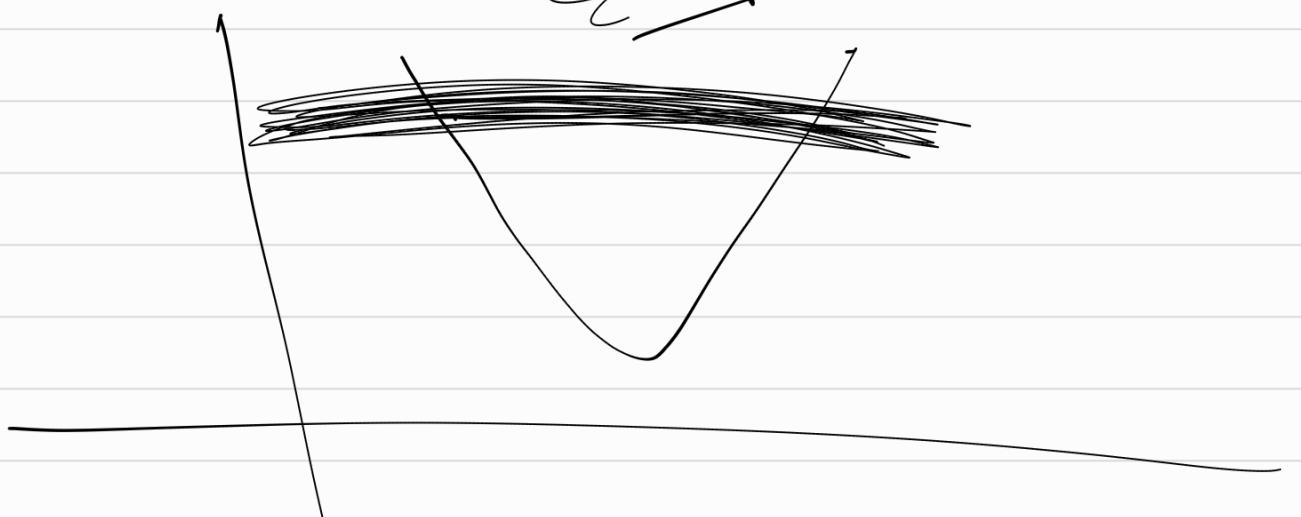
$w_1 -$



$$\left(\frac{0.99}{1.01}\right)^{365} \approx \underline{\downarrow} \underline{\downarrow} \downarrow$$

$$\left(\frac{1.01}{1.01}\right)^{365} = 1.99\underline{1.37}$$

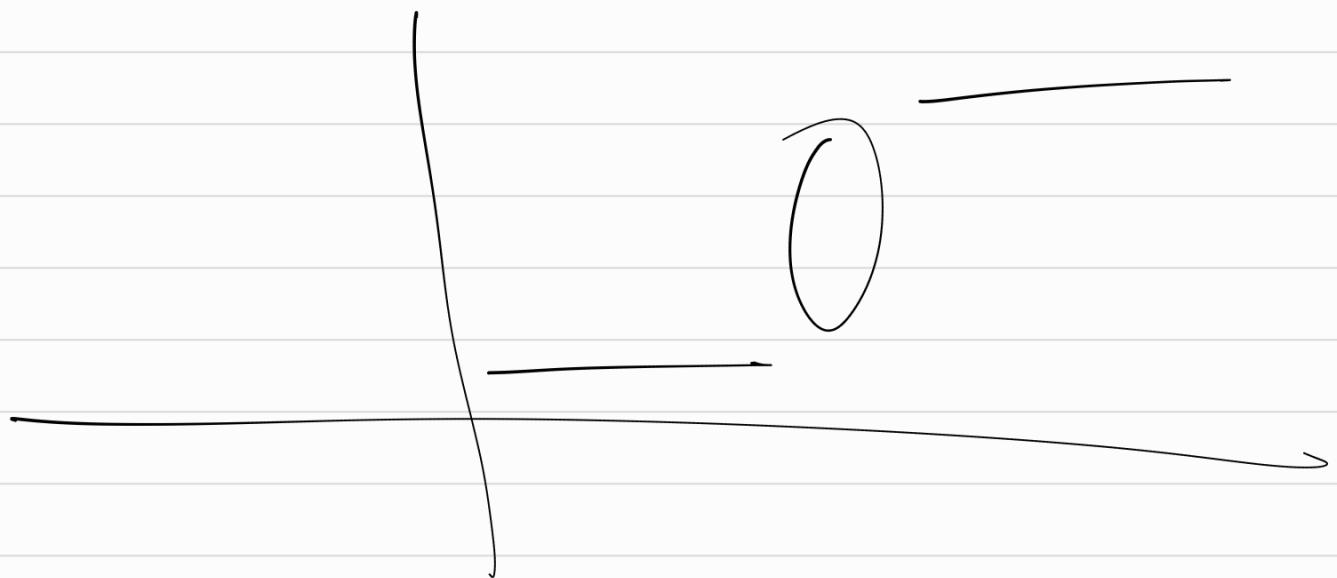
Exploding gradient



$$\hat{y} = w_5 \underline{N_1^3} + w_6 \underline{N_2^2} + b_3$$

$$\frac{\partial \hat{y}}{\partial w_1} = w_5 \times \underline{2 \times N_1}$$

~~(Activation function)~~ → Should be cont & diff.



Activation functions are functions which should be continuous & diff that can be applied at the output of any hidden layers to create non-linear feature or at the 0th to create bounded outputs → Probs

↓ output layer

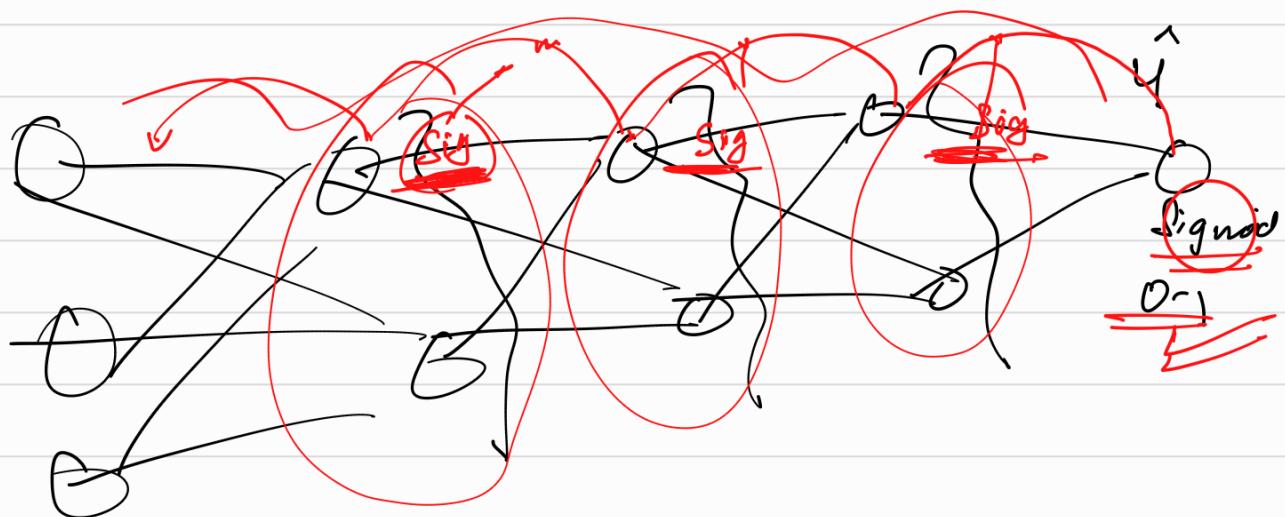
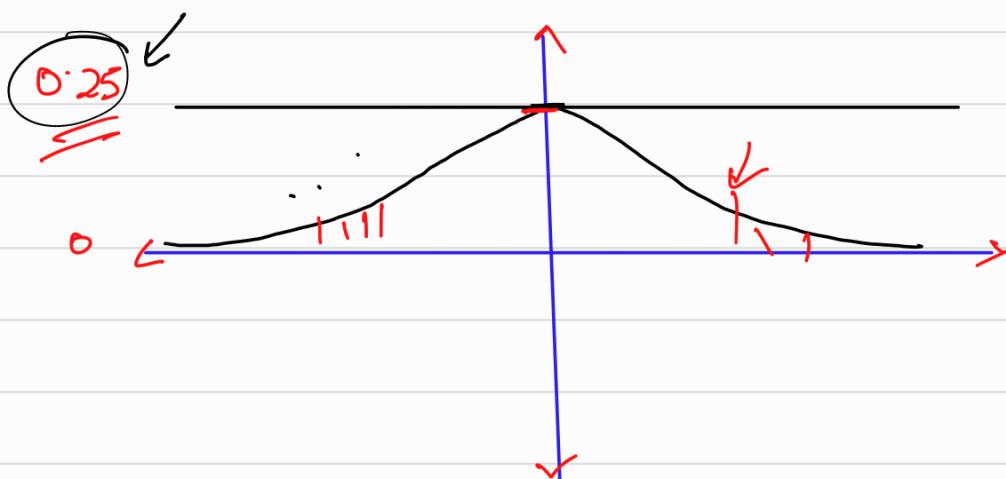
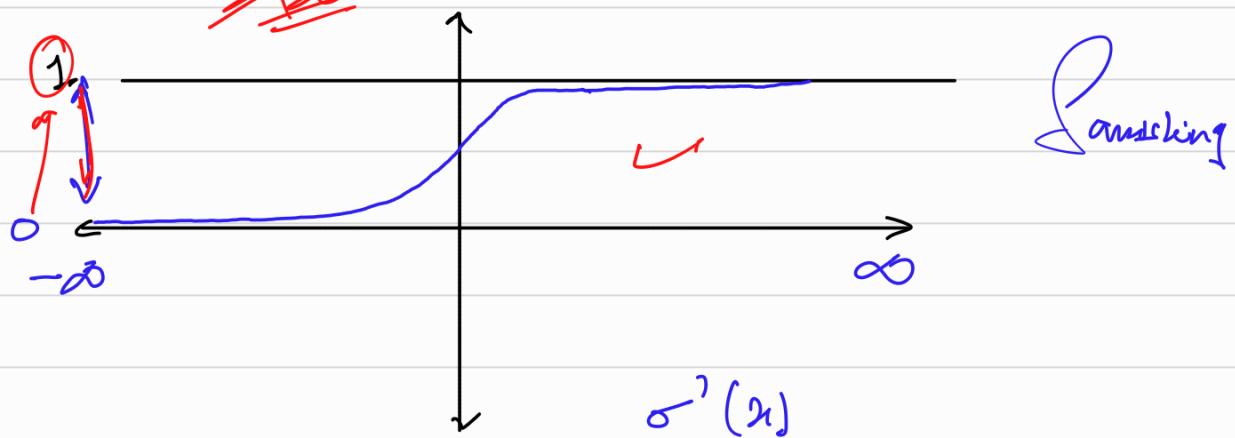
→ Sigmoid

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

$-\infty < x < \infty$

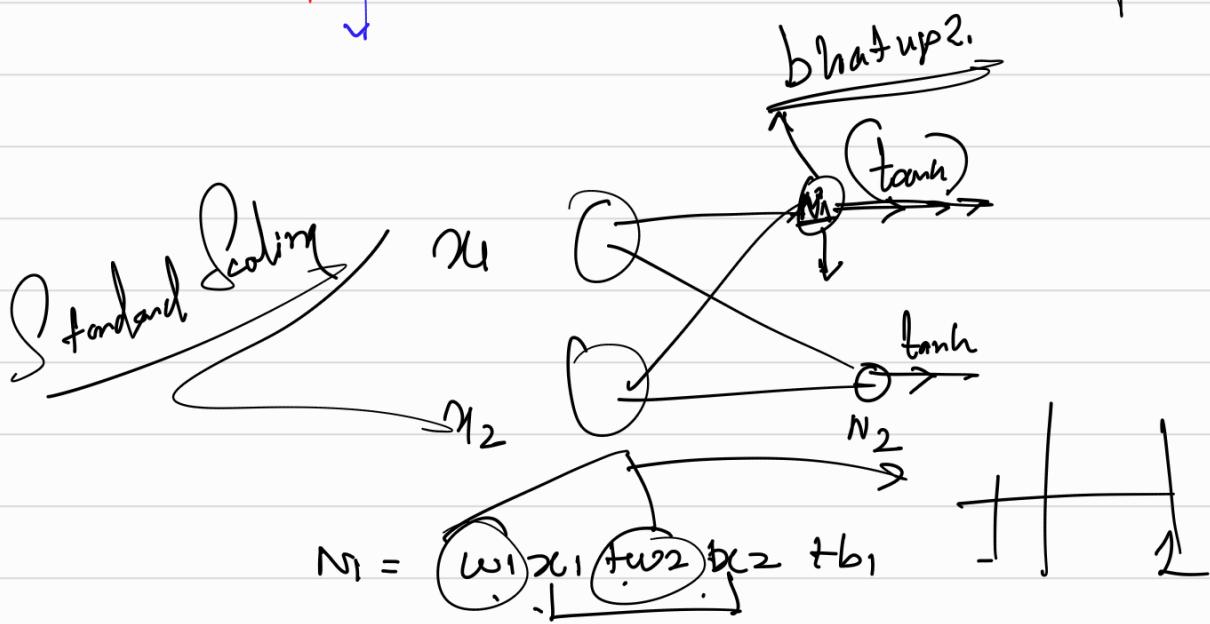
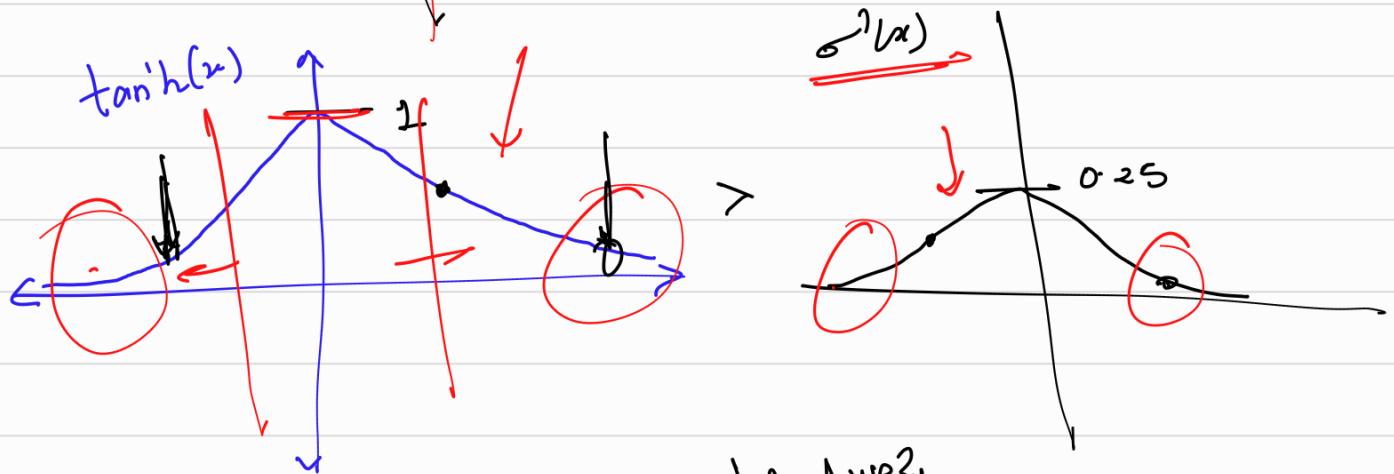
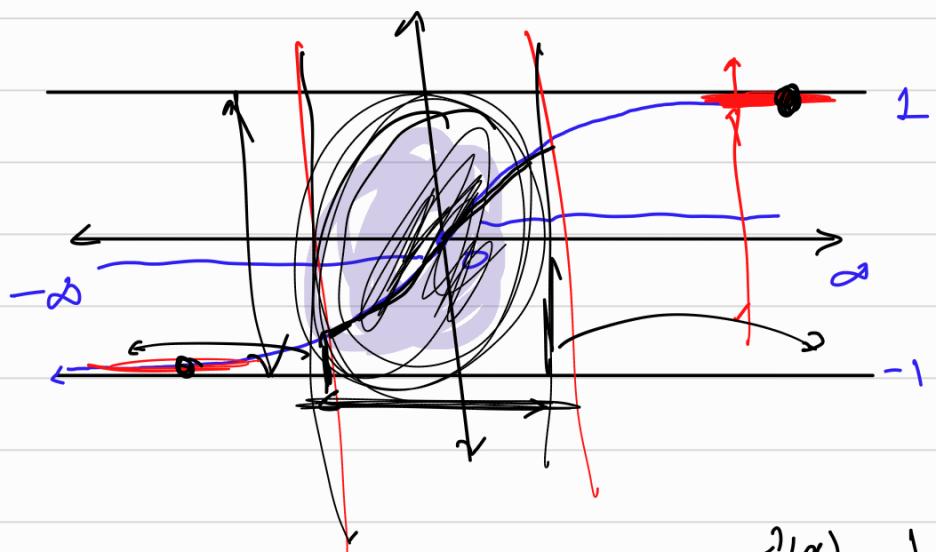
HLX

$0 < \sigma(x) < 1$

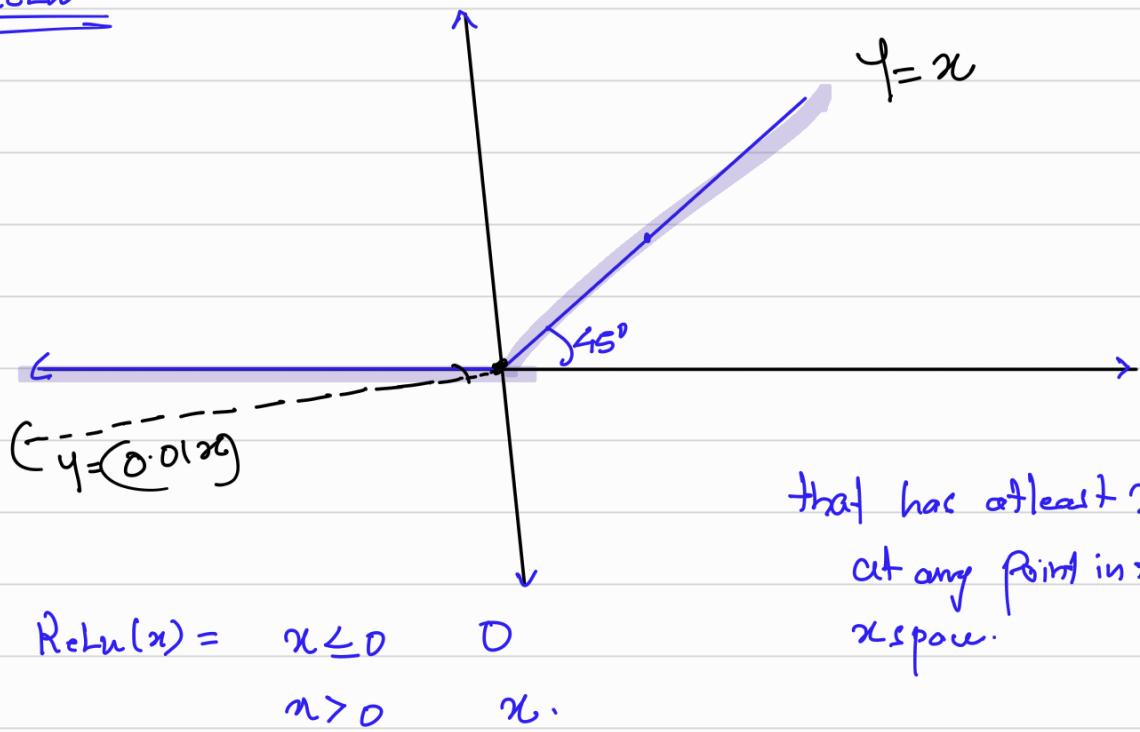


$\tanh \rightarrow$ hyperbolic tangent.

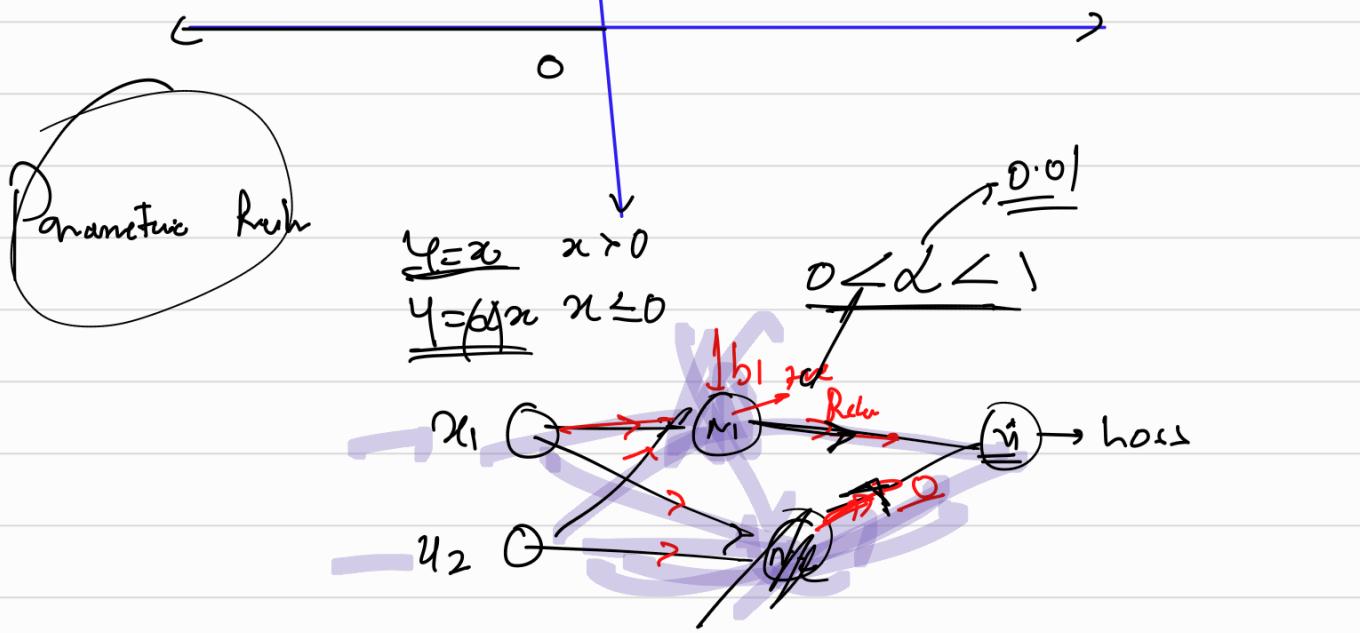
$$\tanh = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



ReLU

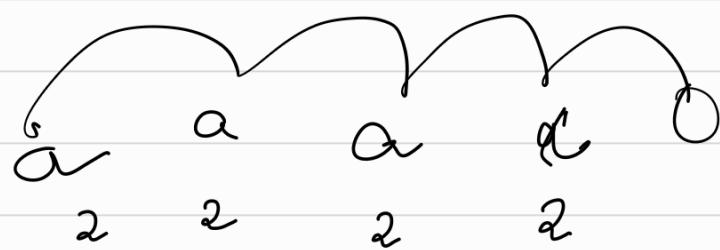


↳
 sketch
 $\text{ReLU}(x) \Rightarrow x \leq 0 \oplus \underline{0.01x}$
 $x > 0 \oplus 1x$



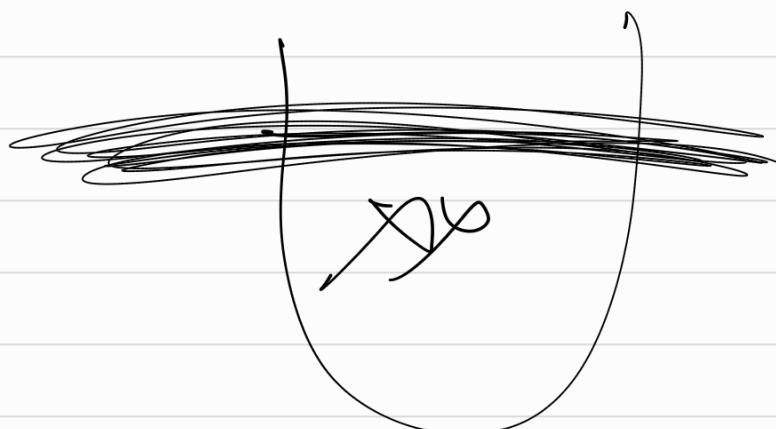
Deep Neural network

$$(2)^{\cancel{0} + \cancel{10}} = 32$$


 a a a a
 2 2 2 2

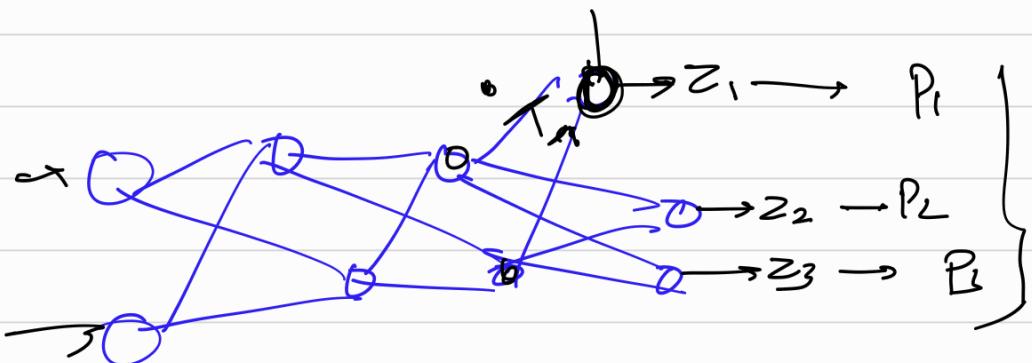
$$(2)^3 \quad \underline{(2)^{100}}$$

$$\frac{dh}{d\alpha} = w_r - \alpha \left(\frac{1}{g_m} \right)$$



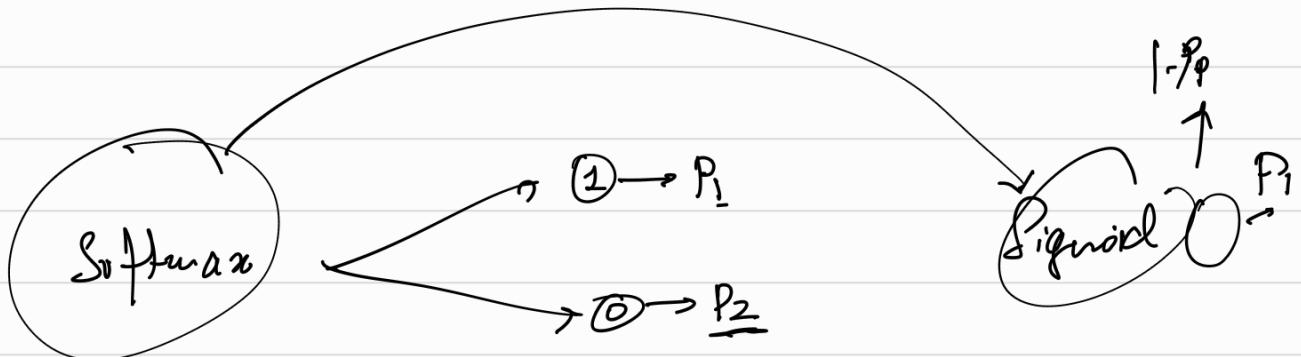
Softmax

: needs to be applied at the o/u of a multiclass problem



$$P_1 + P_2 + P_3 = 1$$

$$P_1 = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}} \quad P_2 = \frac{e^{z_2}}{e^{z_1} + e^{z_2} + e^{z_3}} \quad P_3 = \frac{e^{z_3}}{e^{z_1} + e^{z_2} + e^{z_3}}$$



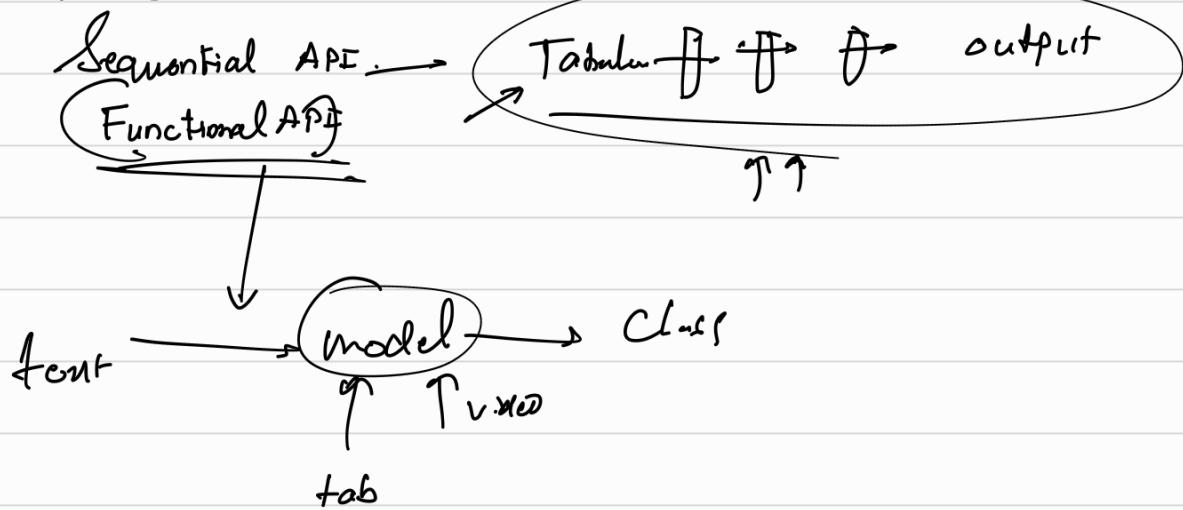
$$P_1 + P_2 = 1$$

$$P_1 = 1 - P_2$$

$$P_2 = 1 - P_1$$

Bocak

Lecture - 5:



Specce.
 $y \rightarrow LE(0, 1, 2)$
 $(y=1)$

$\hat{y} = [0.1, 0.3, 0.6]$

$-\log(\hat{y}[y])$

$-\log 0.3$

$y \rightarrow DCE.$
 $y \rightarrow [0, 1, 0].$

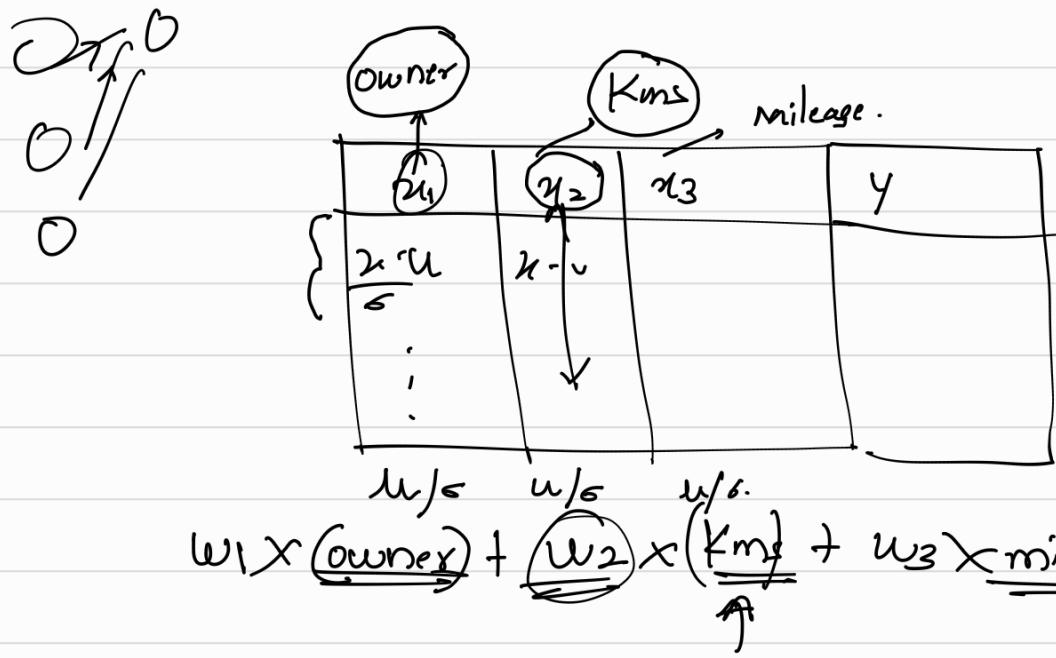
cce.

$\hat{y} = [0.1, 0.3, 0.6]$

$y = [0, 1, 0].$

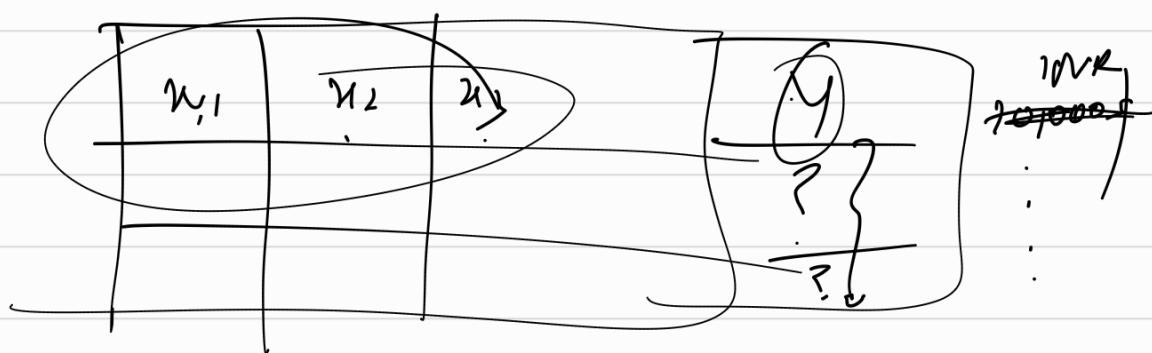
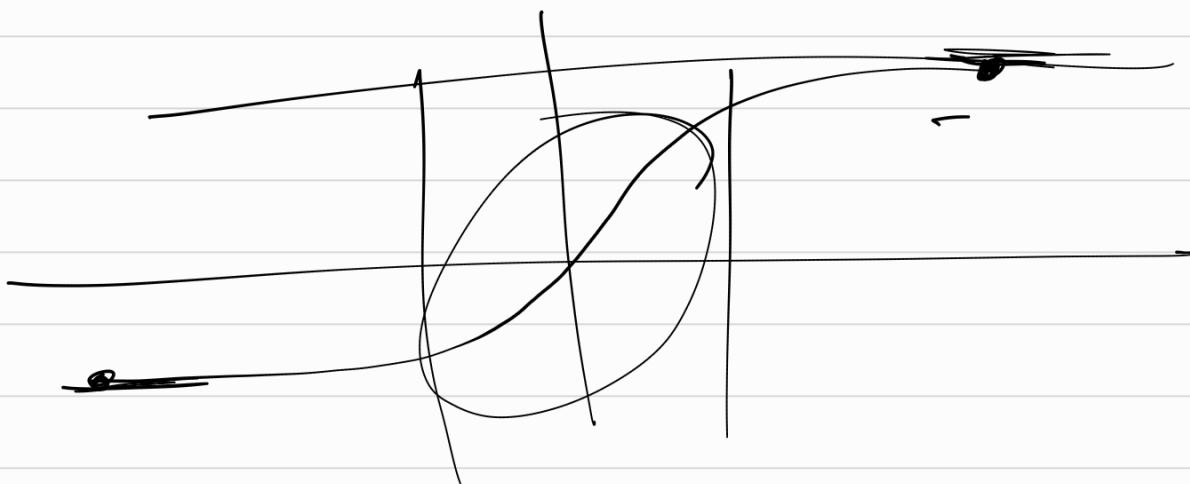
$-[0 \times \log 0.1 + 1 \times \log 0.3 + 0 \times \log 0.6]$

$= -\log 0.3$



(1) Honda City
50,000 ↘

(2) Honda City
40,000



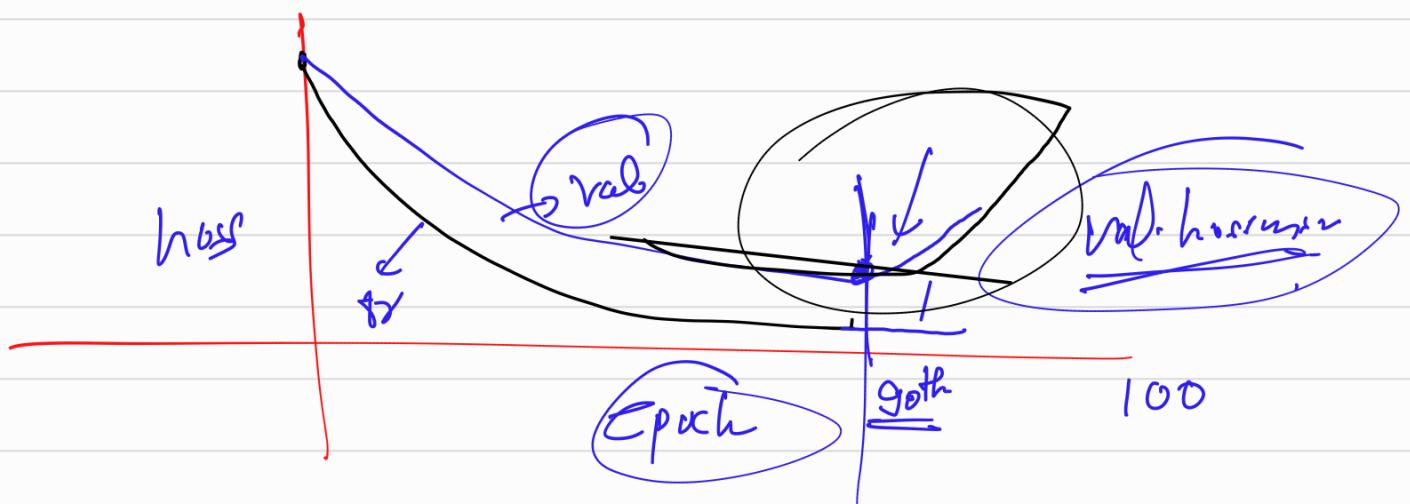
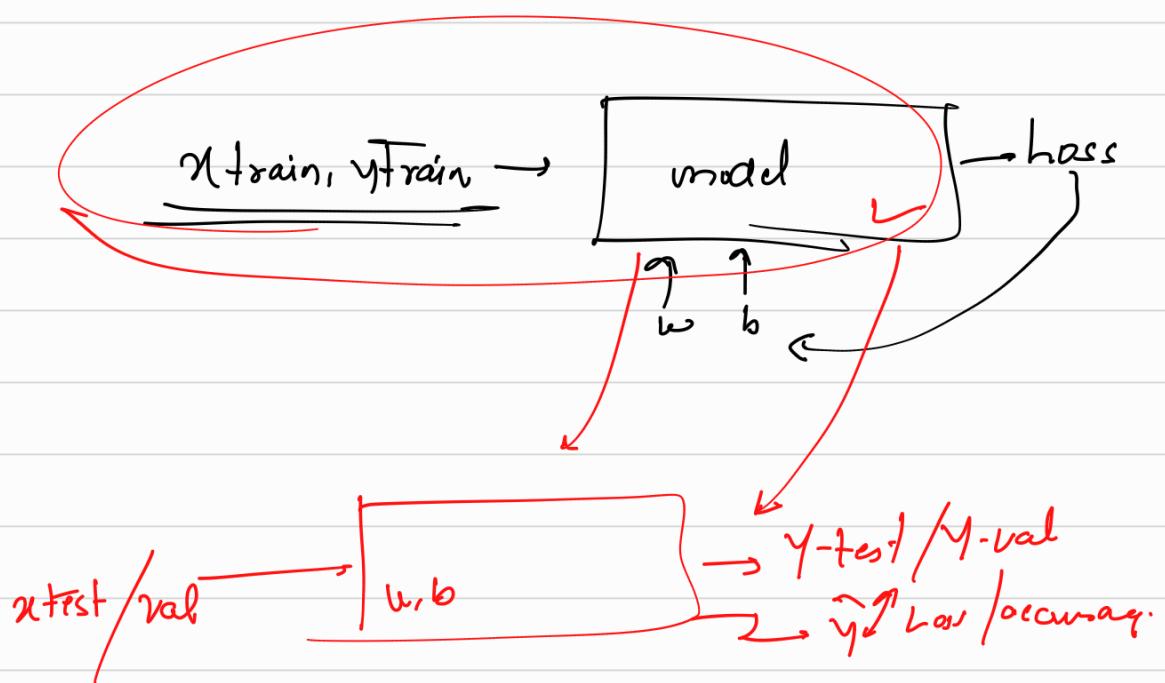
3,-3
($w_2 \sim -1, 1$)

$\begin{matrix} 3,-3 \\ 3,1 \\ -1 \end{matrix}$

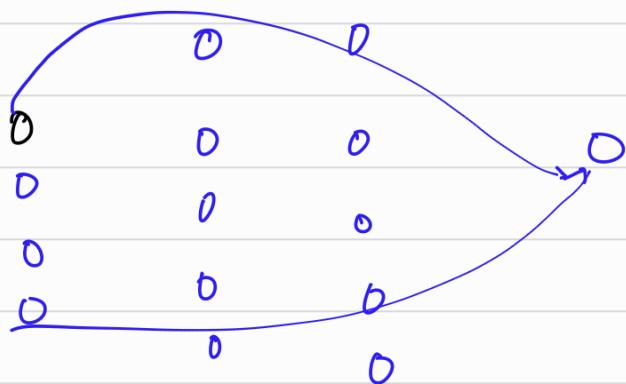
$2-1 \rightarrow$

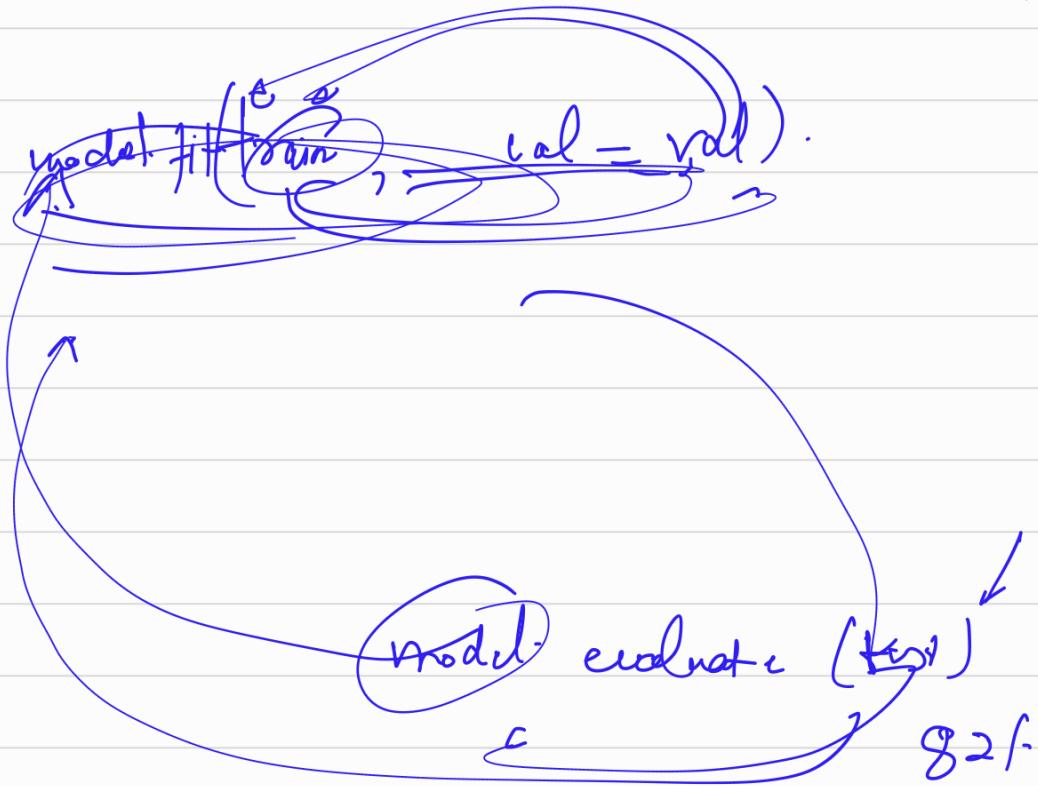
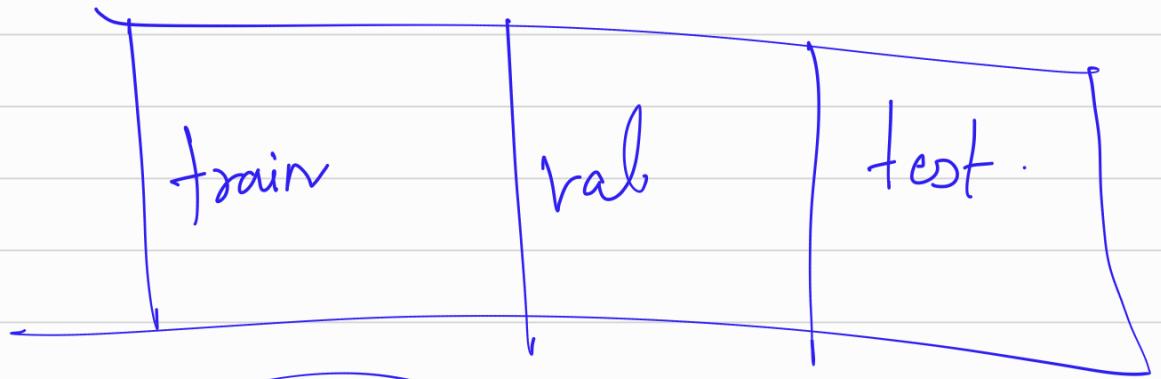
$0 \rightarrow \underline{\underline{50,000}}$

$L = (\underline{\underline{50,000 - 0.9}})^2$



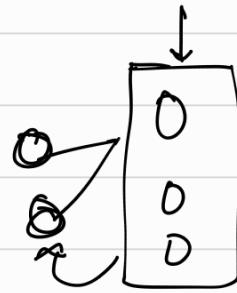
Callbacks



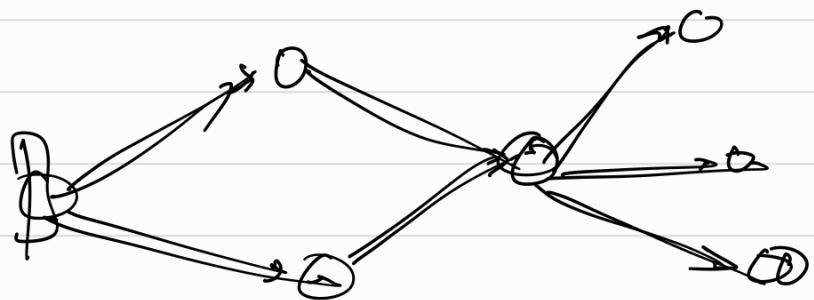


Functional API:

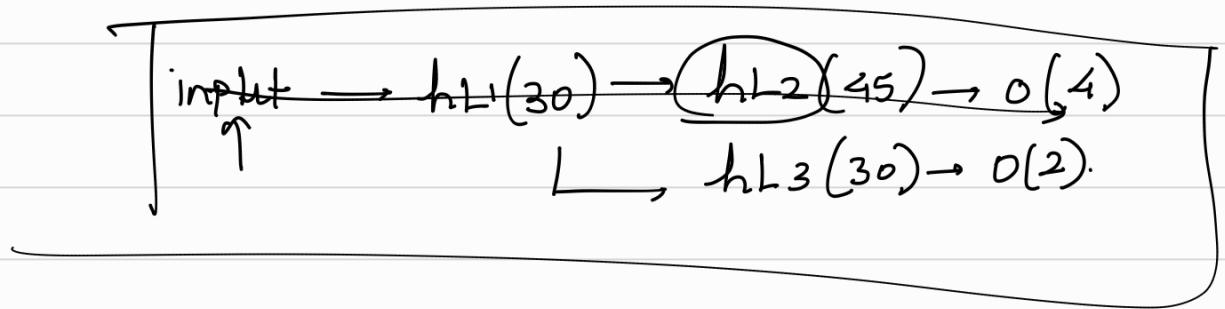
Graph Structure



any layer (ϵ)



$D \rightarrow D$



Input $\xrightarrow{(4)} hL_1(30) \rightarrow hL_2(40)$

Case 1: Input $\xrightarrow{(4)} hL_1(20) \rightarrow hL_2(2) \rightarrow 50$

Case 2:

inp1 = Input(4,)

inp2 = Input(4,)

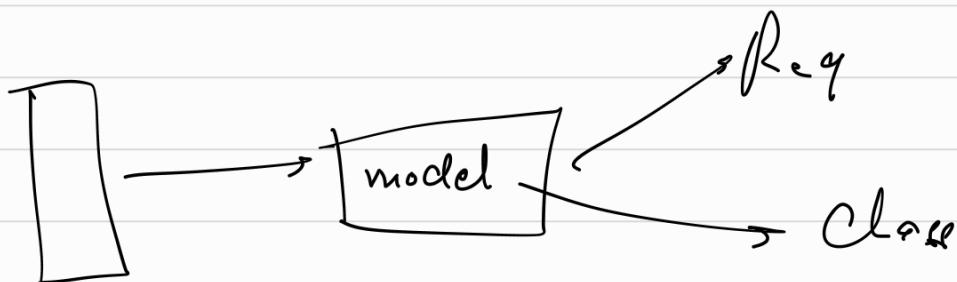
hL1 = Dense(30)(inp1)

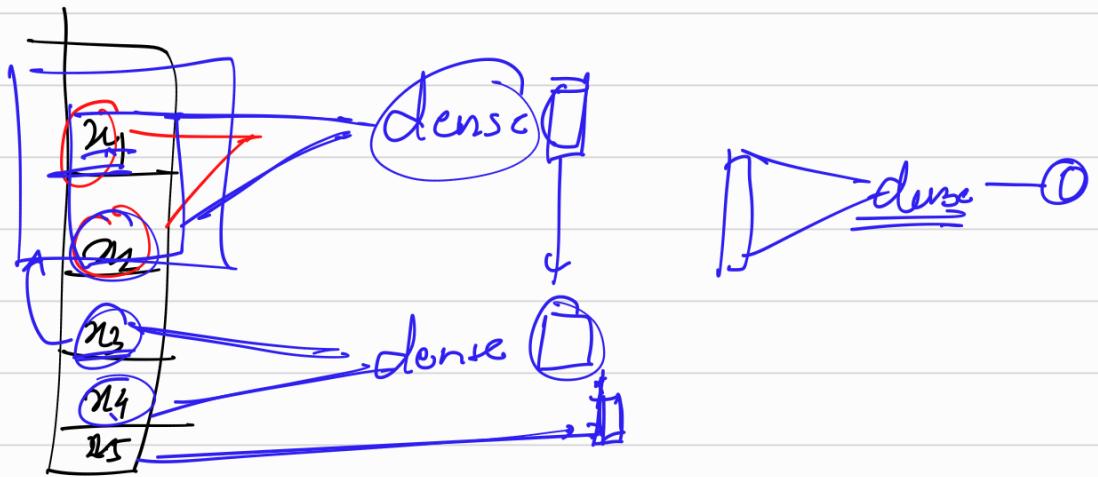
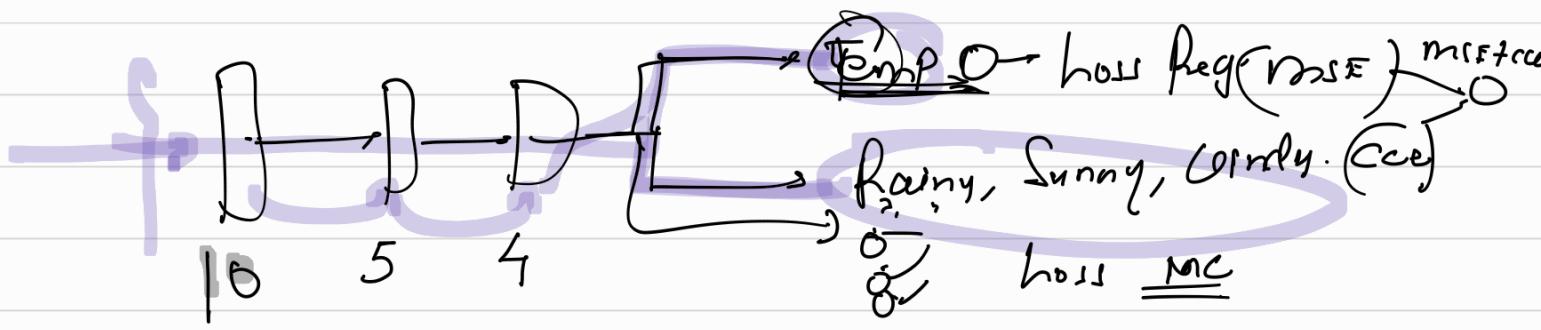
hL2 = Dense(40)(hL1)

hL3 = Dense(20)(inp2)

hL4 = Dense(50)(hL3)

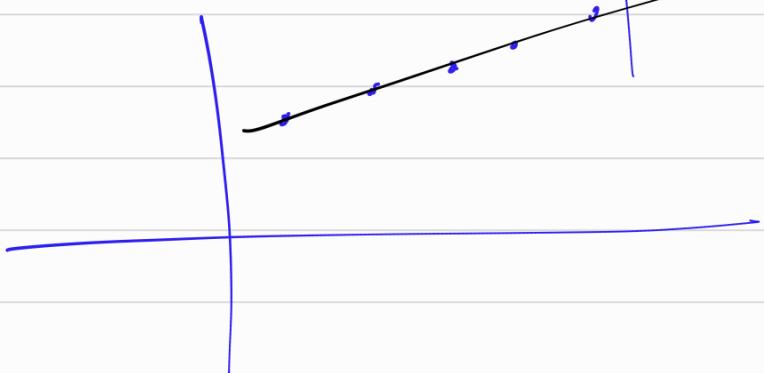
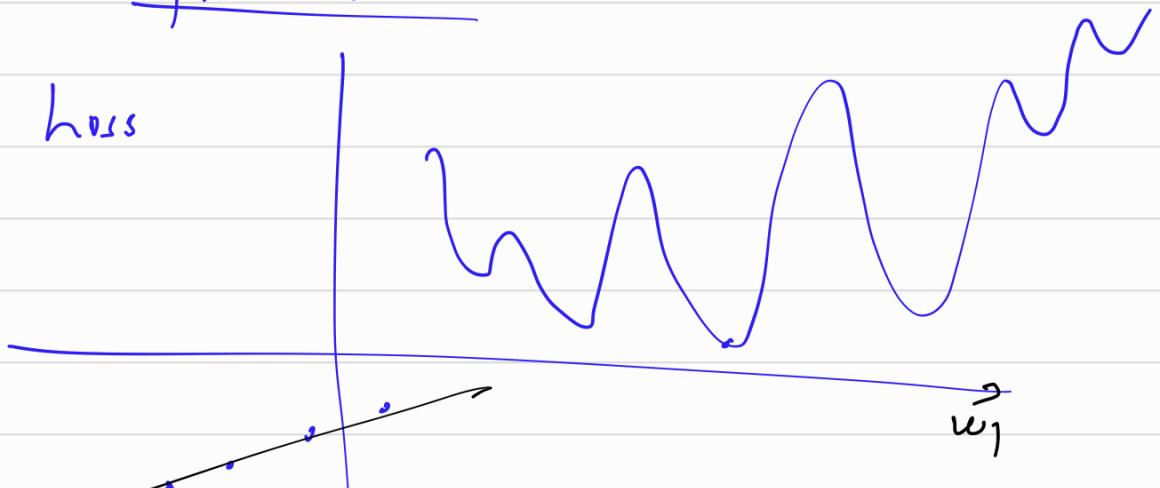
O = Concatenate([hL2, hL4])

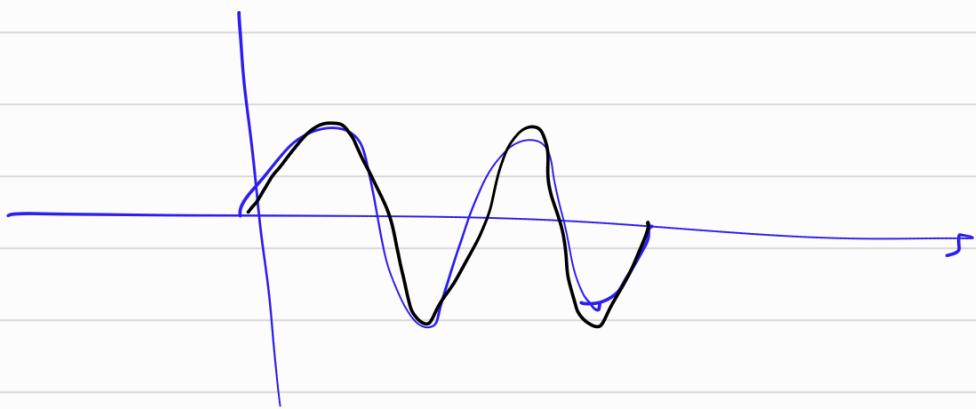
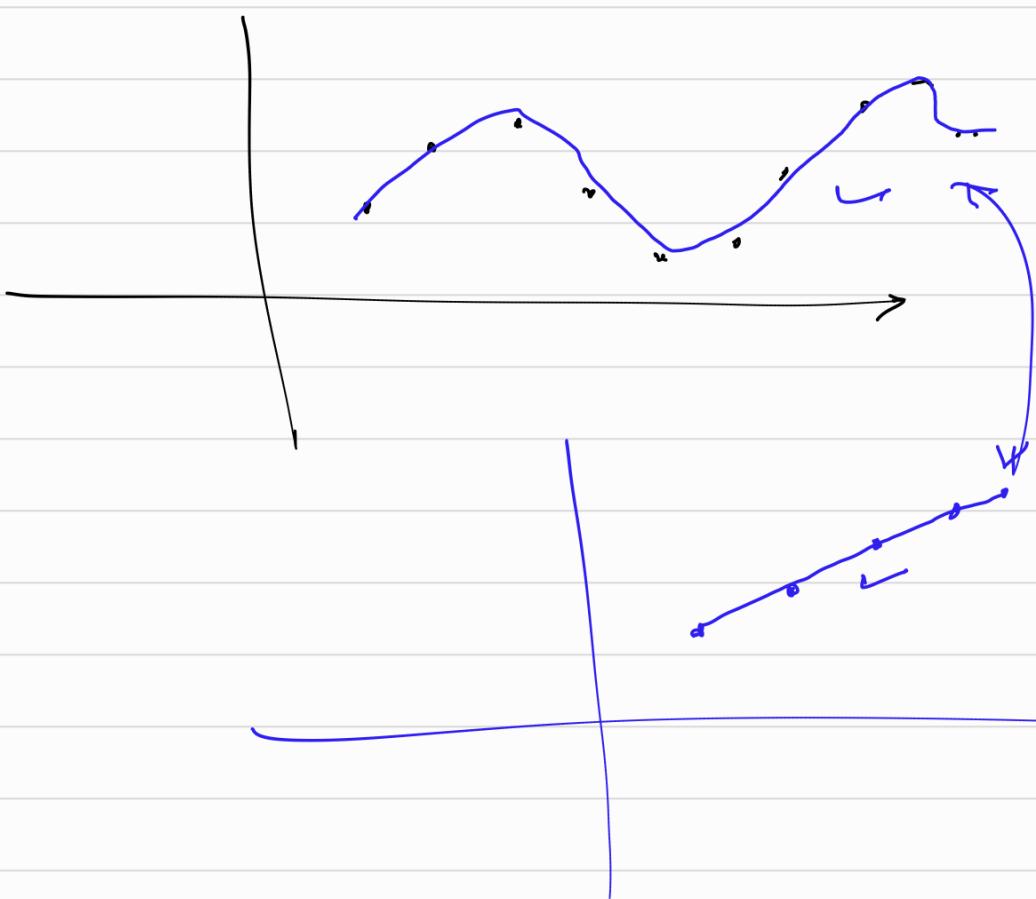
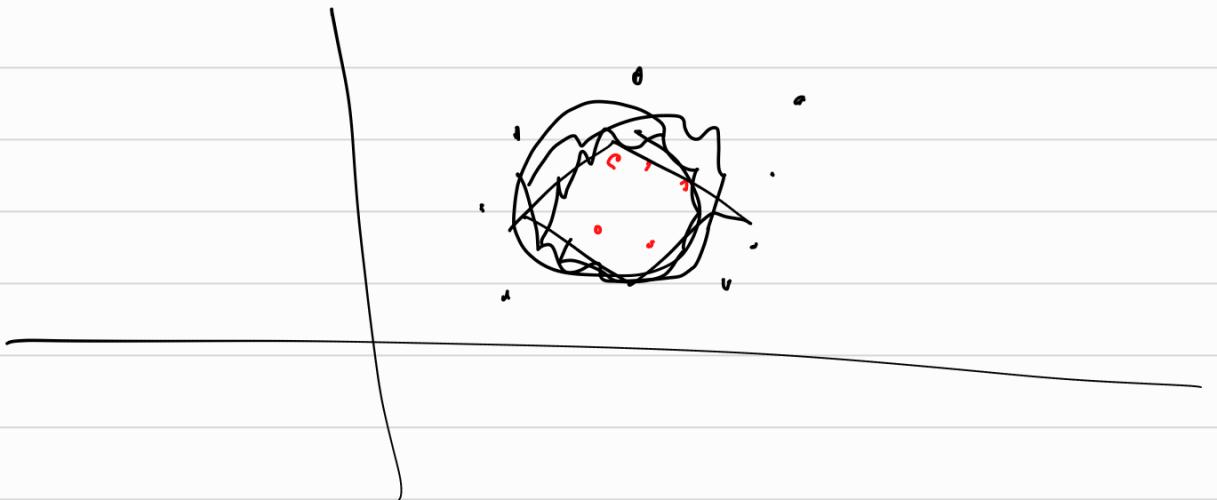




Optimizers

loss





gg. gggg).



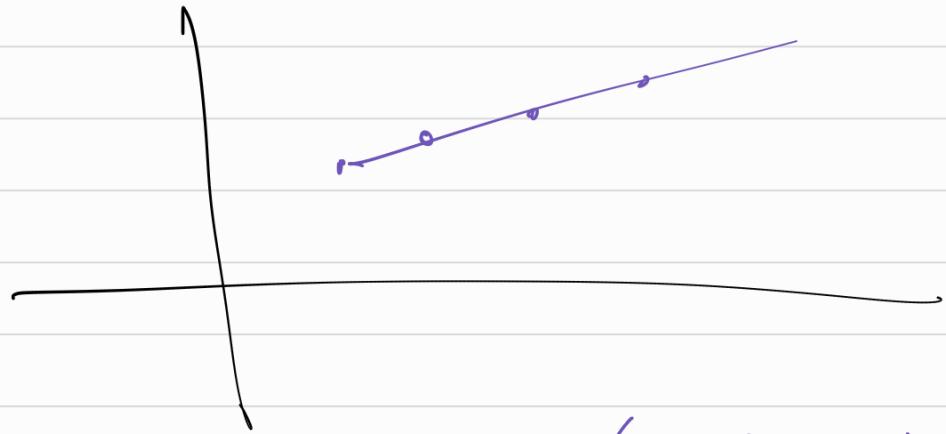
(Regularization.)

- Regularization
- Dropout
- Optimizers.
- Callbacks
- Batch Normalization
- weight initialization

Hyperparameter Tuning

Regularization:

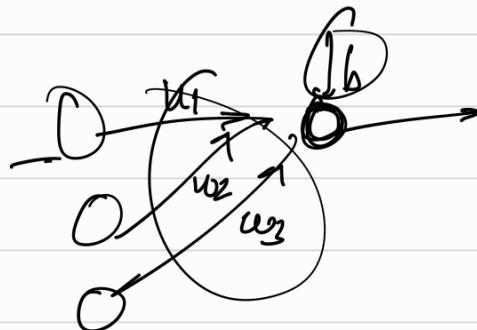
↓ ↓
(Ridge and Lasso.) ?



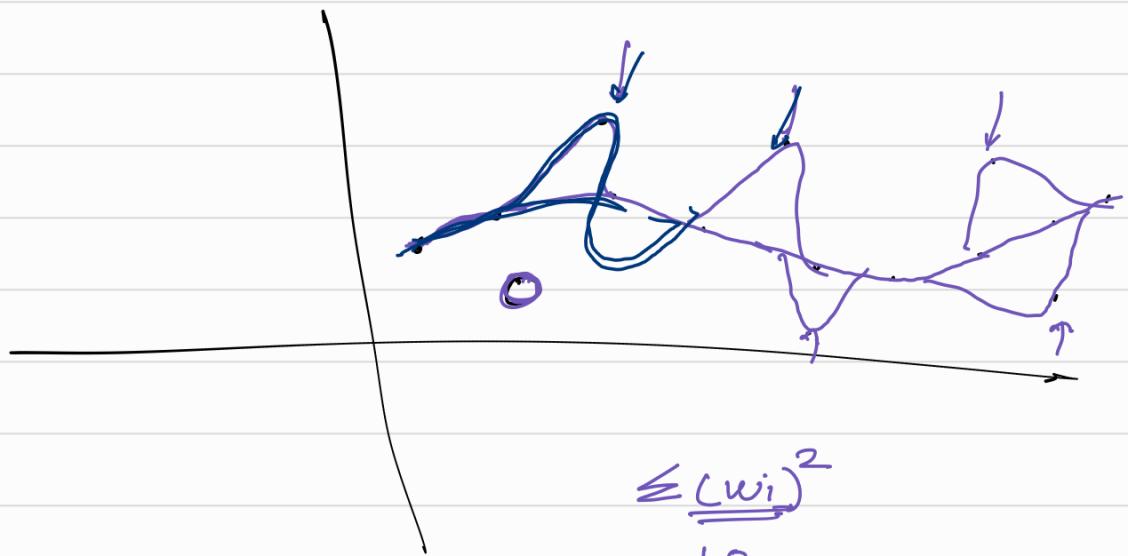
Overfit → has high variance (model is extremely sensitive to changes in n data).

$$\textcircled{1} \quad y = \underline{100x_1} + \underline{500x_2} \quad y=1000$$

$$\textcircled{2} \quad y = 2x_1 + x_2 = \textcircled{2}$$



$$N1 = \underline{w_1} \underline{x_1} + \underline{w_2} \underline{x_2} + \underline{w_3} \underline{x_3} + b_1$$



TL mSE

$$510 = 500$$

$$425 = 400$$

$$375 = 320$$

$$355 = 260$$

$$320 = 200$$

$$305 = 150$$

$$\underline{\underline{345 = 145}}$$

$$\leq \frac{(w_i)^2}{10}$$

$$25$$

$$55$$

$$95$$

$$= 120$$

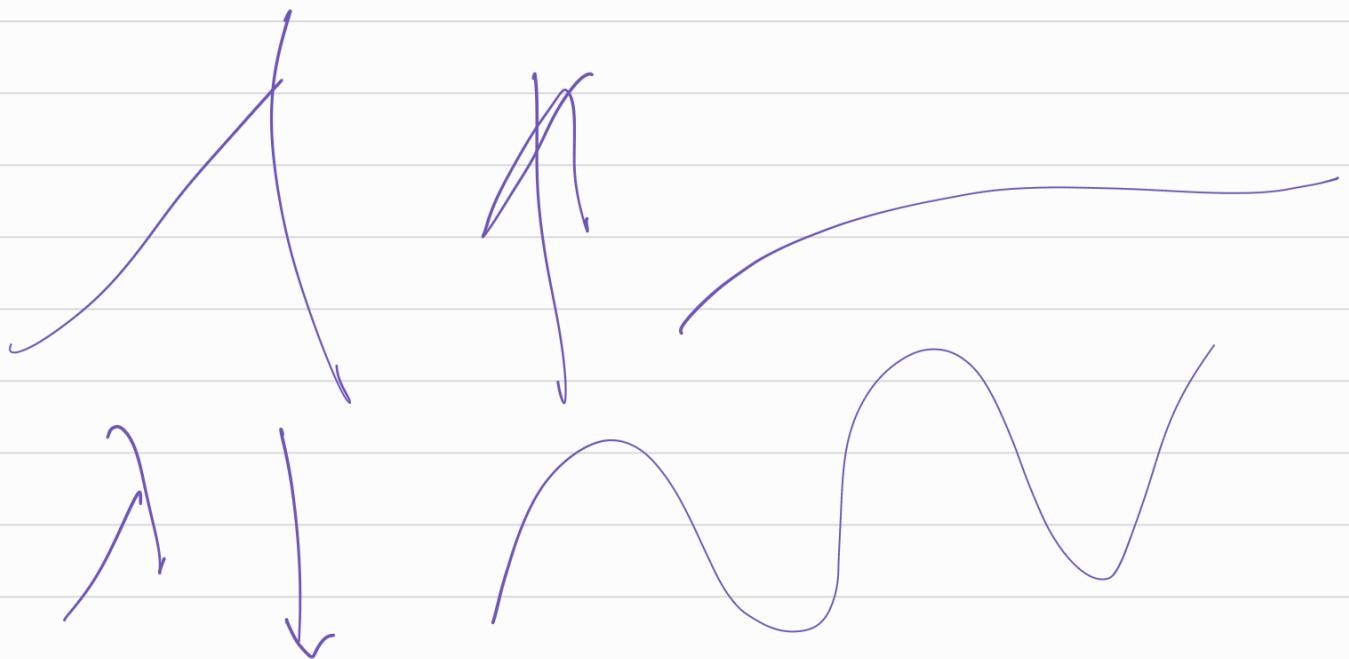
$$= \frac{155}{200} \}$$

$$TL = \text{mse} + \boxed{\frac{1}{g}} \sum_{j=1}^p w_j^2$$

Lagrangean multiplier

↓
Hyperparameter (0 - ∞)

↓
~~Val-loss~~



$\lambda =$

$\lambda = 0.5$

$\lambda = 0.9$

$\lambda = 0.95$

$\lambda = 0.5$

$\rightarrow 92\%$

$\rightarrow 95\%$

Reg.

$$T_L = \text{mse} + \lambda \sum w_j^2 \quad \text{Ridge } (L^2).$$

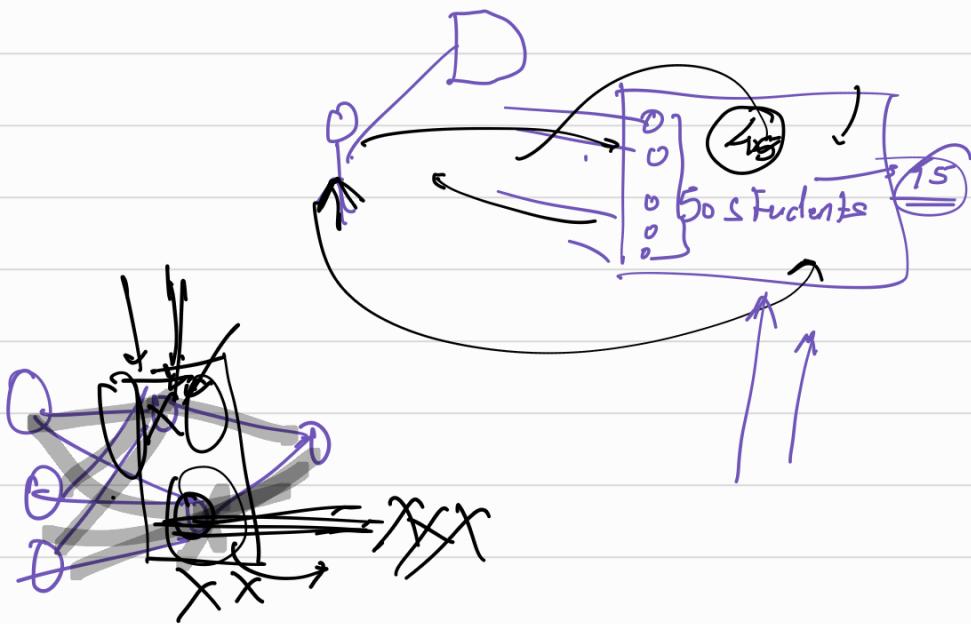
$$T_h = \text{mse} + \lambda \sum |w_j| \quad \text{Lasso } (L_1).$$

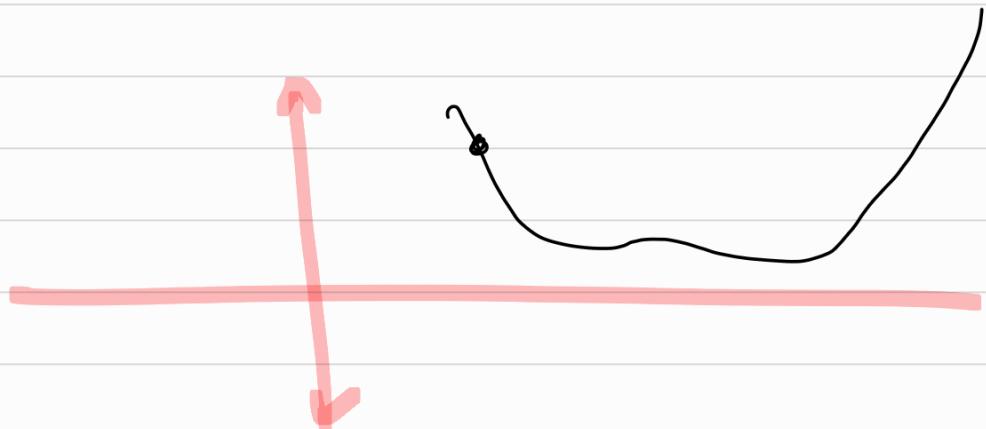
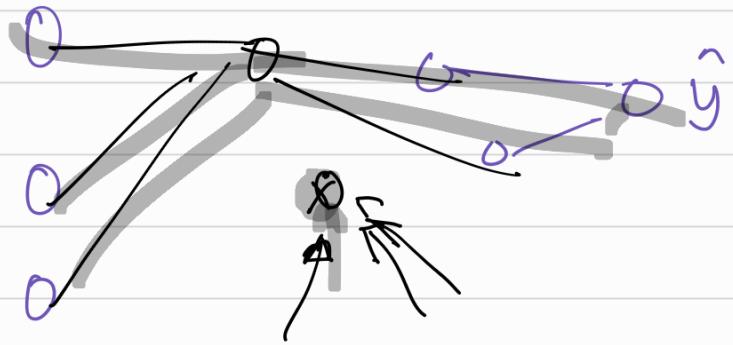
$$T_L = \text{mse} + \lambda_1 (\sum w_j^2) + \lambda_2 \sum |w_j| \quad L_1 L_2$$

Elastic Net

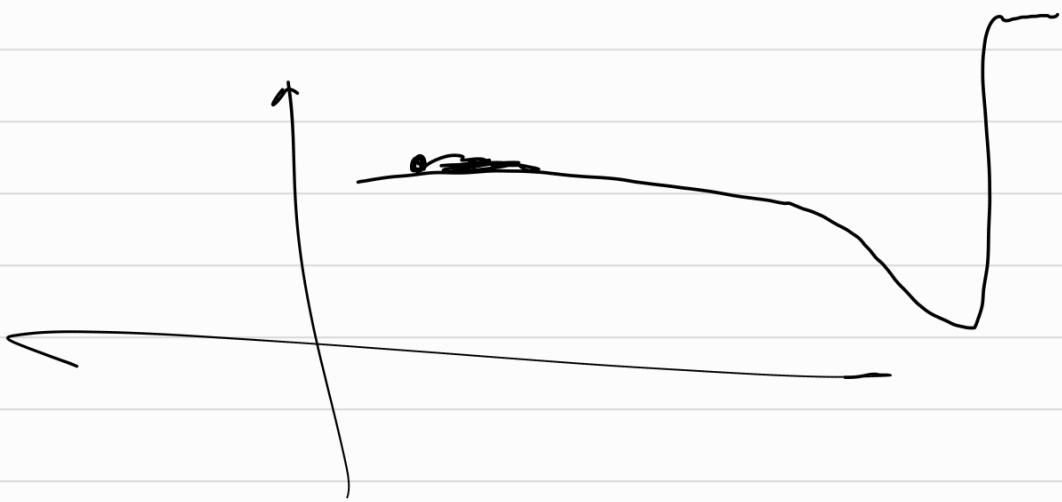
Break!

Dropout:



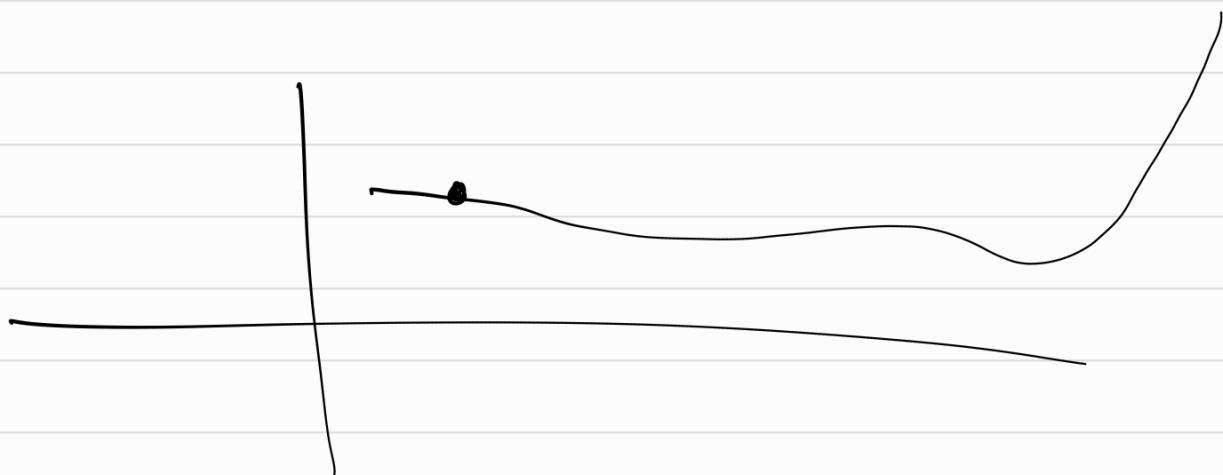
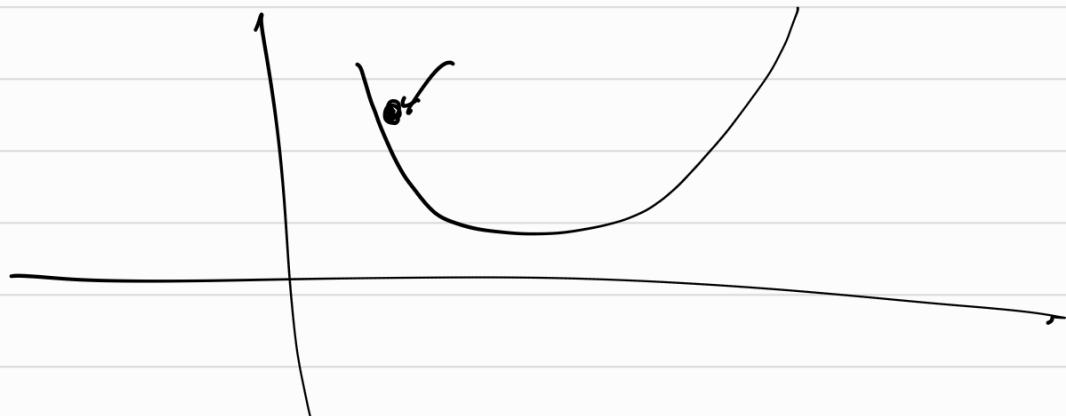
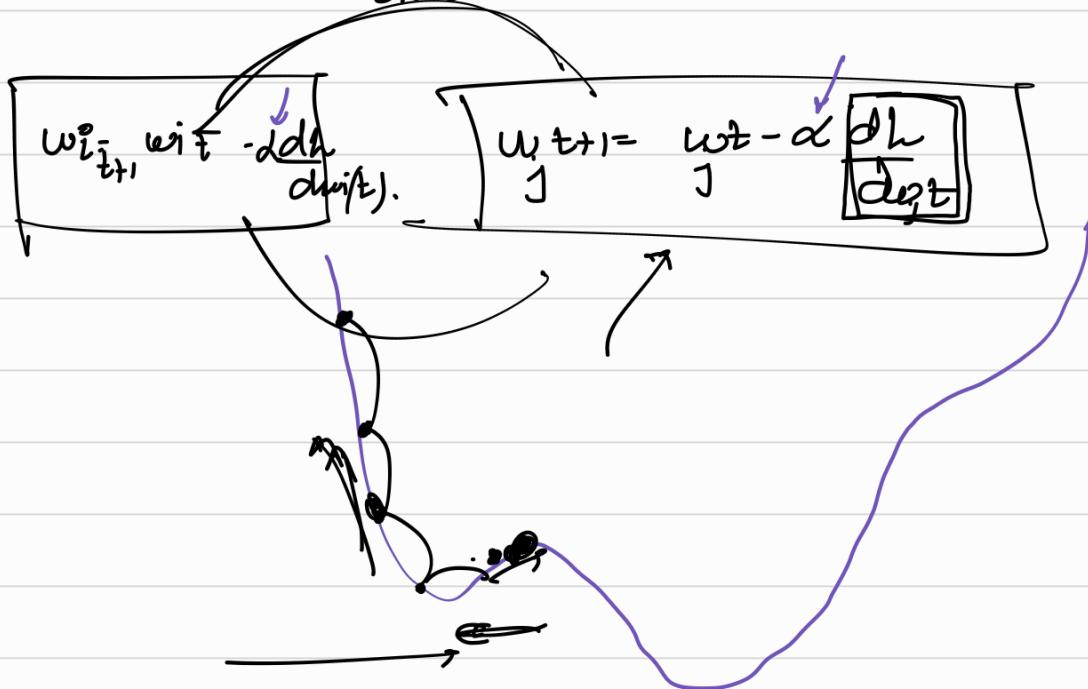


10

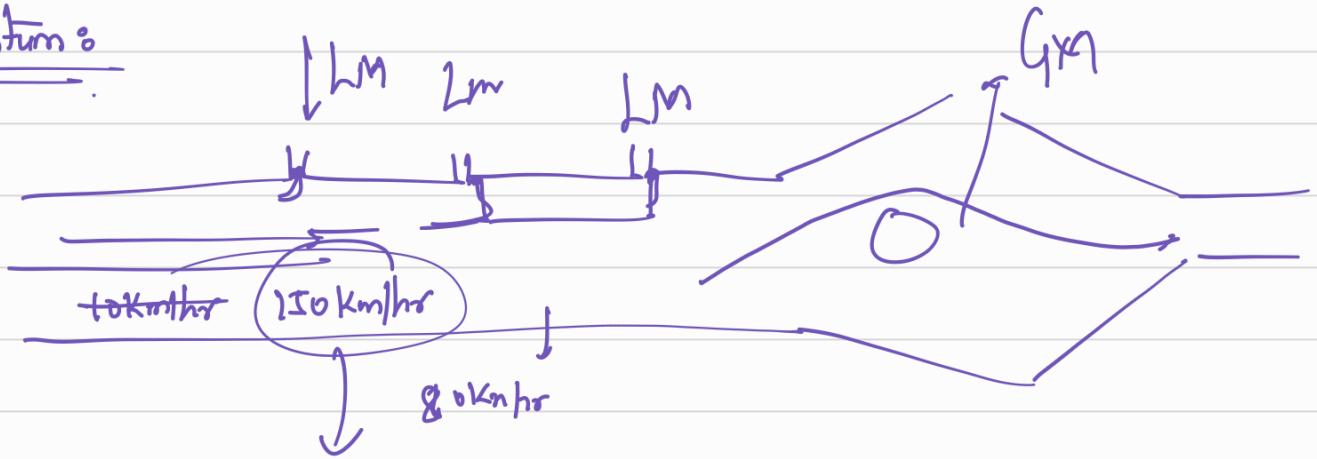


optimizers :

minimizing the loss by changing the weights and biases.



Momentum :



Momentum

GD

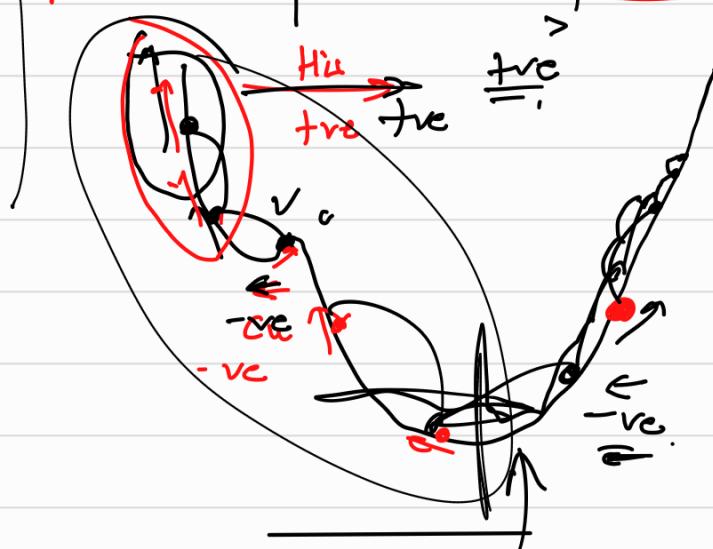
$$w_{t+1} = w_t - \alpha \frac{\partial L}{\partial w_t}$$



GD with momentum

$$w_{t+1} = w_t - \text{update}_t$$

$$\text{update}_t = \beta \text{update}_{t-1} + (1-\beta) \frac{\partial L}{\partial w_t}$$



$t=0, t=1, t=2, \dots$

$$\text{update}_1 = (1-\beta) \alpha \frac{\partial L}{\partial w_1}$$

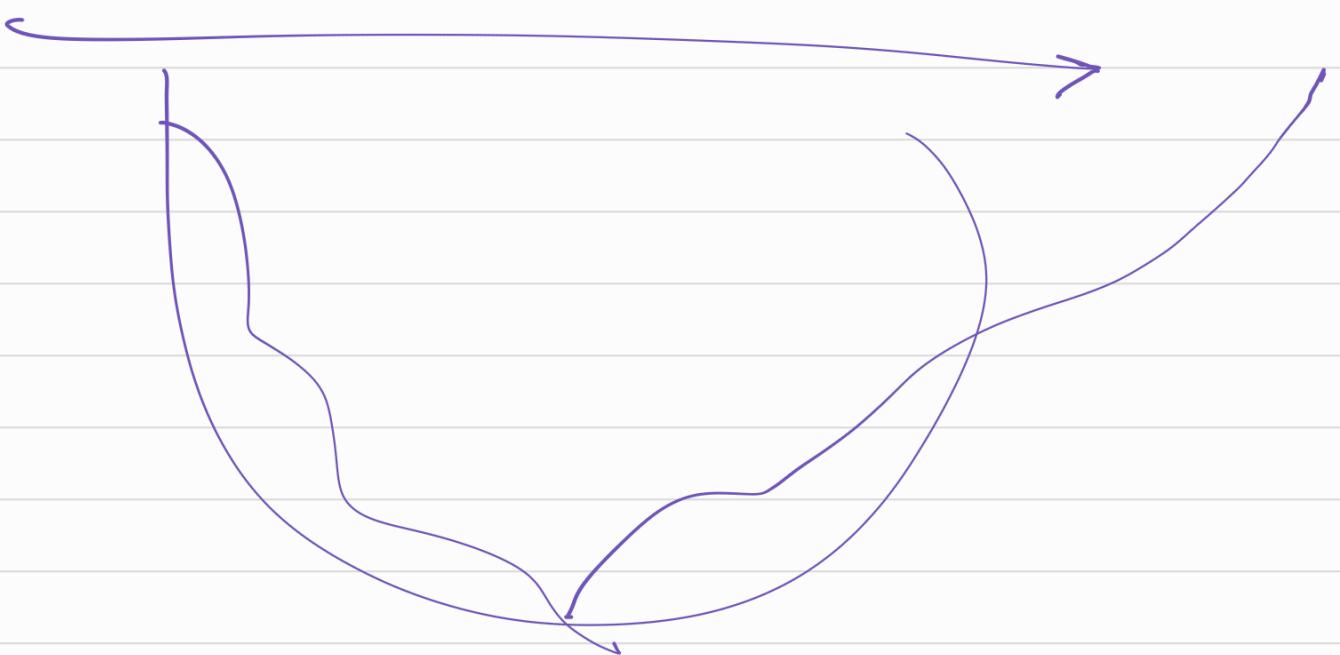
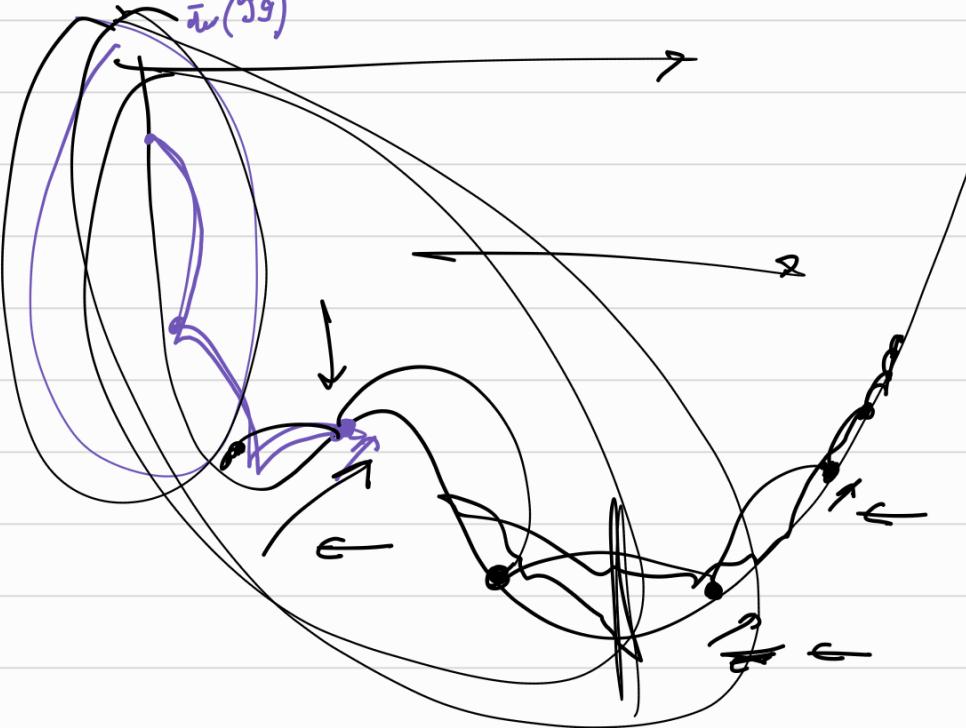
$$\text{update}_2 = \beta \text{update}_1 + (1-\beta) \alpha \frac{\partial L}{\partial w_2}$$

$$\text{update}_2 = \beta \left((1-\beta) \alpha \frac{\partial L}{\partial w_1} \right) + (1-\beta) \alpha \frac{\partial L}{\partial w_2}$$

$$\text{update}_3 = \beta \text{ update}_2 + (1-\beta) \alpha \frac{\partial L}{\partial w_3}$$

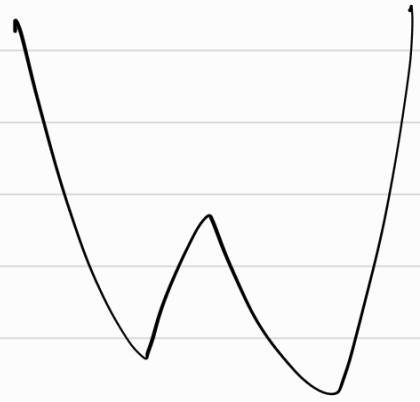
$$w_3^{update} = \beta \left(\beta (1-\beta) \alpha \frac{\partial h}{\partial w_1} + (1-\beta) d \frac{\partial h}{\partial w_2} \right) + (1-\beta) \alpha \frac{\partial h}{\partial w_3}.$$

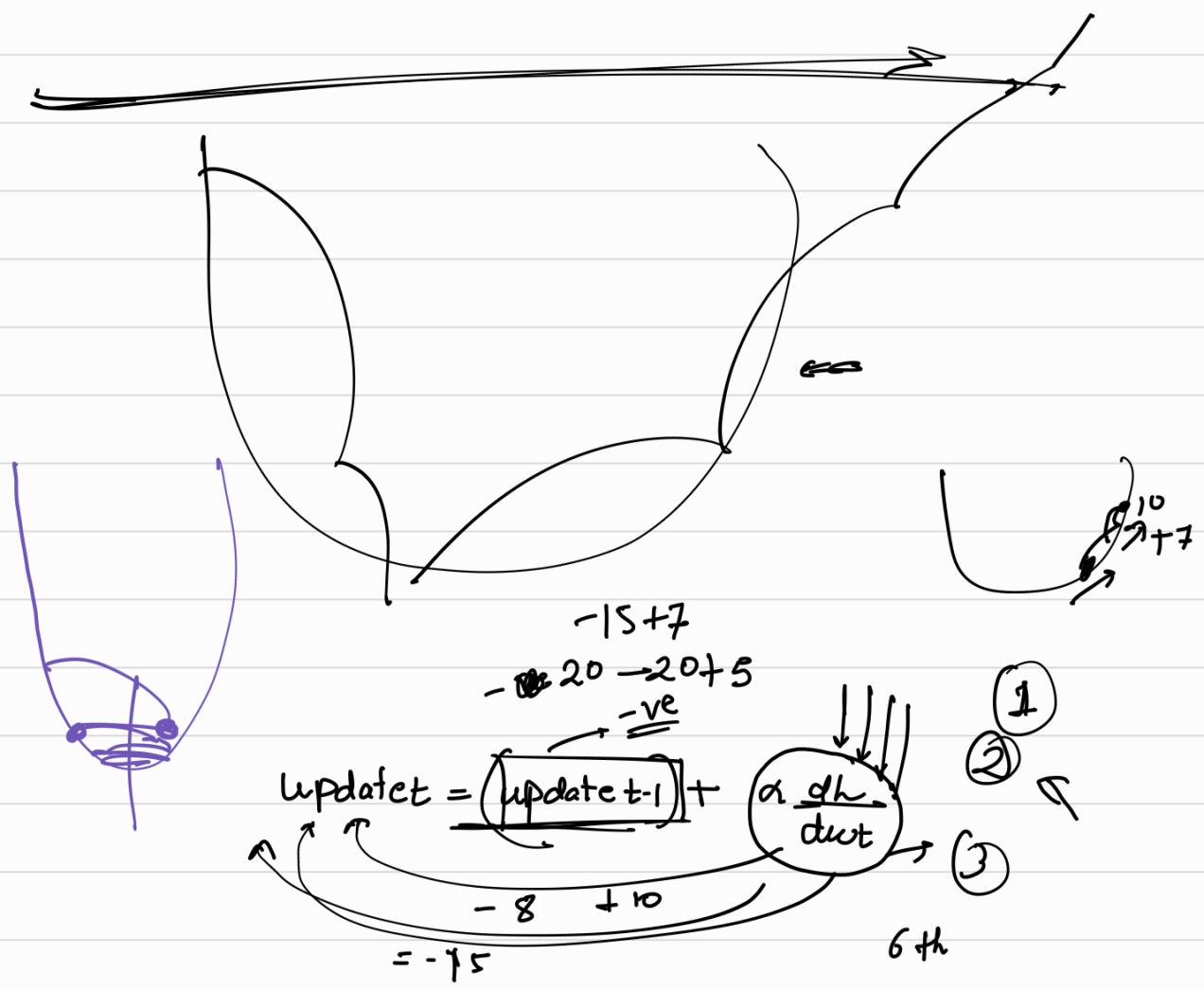
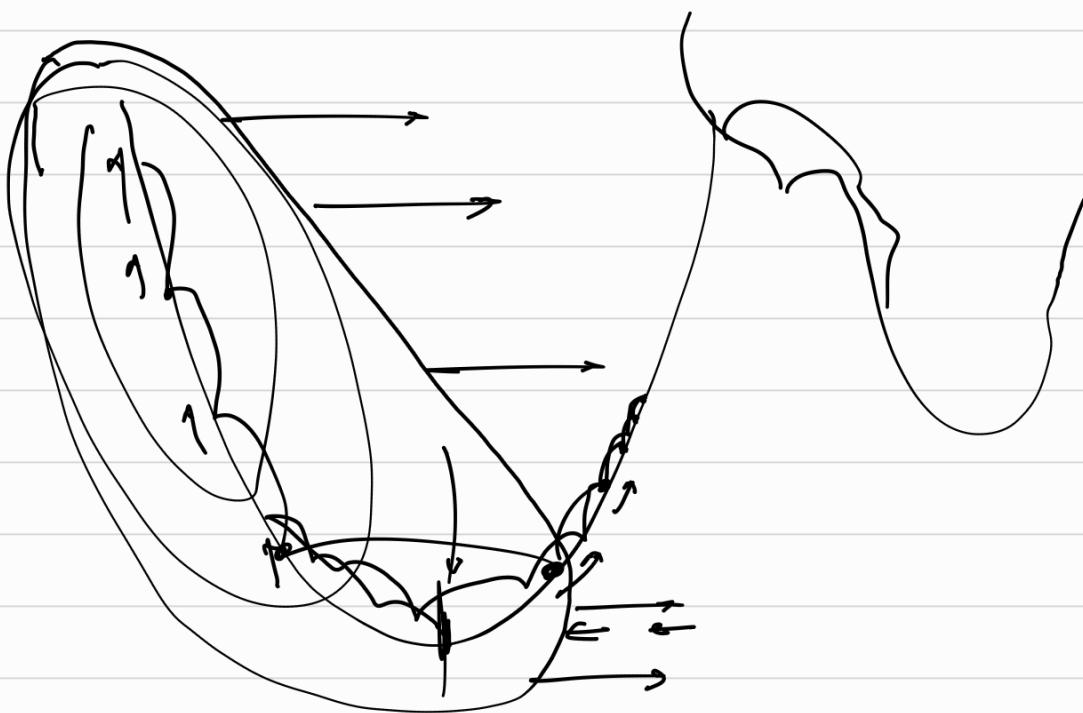
$$\text{update } z_1 = \underbrace{\beta^2}_{B < 1} (1 - \beta) \frac{\partial h}{\partial w_1} + \frac{\beta}{2} (1 - \beta) \frac{\partial^2 h}{\partial w_1^2} + \frac{(1-\beta)}{2} \frac{\partial^2 h}{\partial w_2^2} + \frac{(1-\beta)}{2} \frac{\partial^2 h}{\partial w_3^2}$$

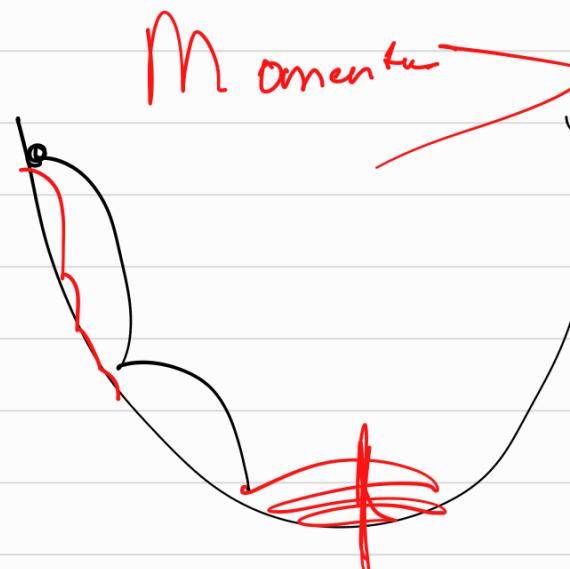
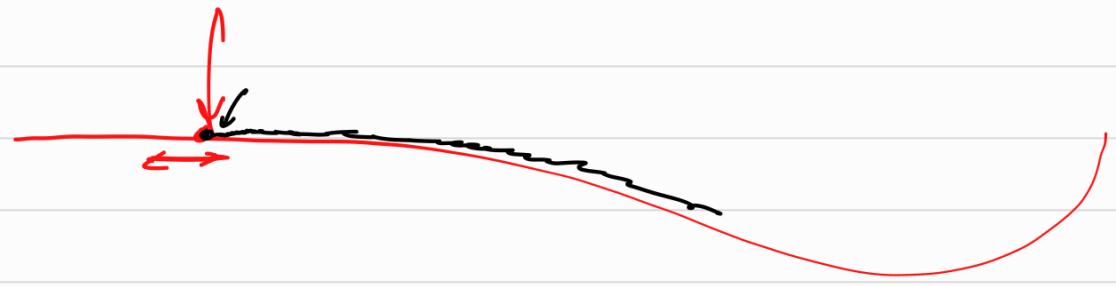
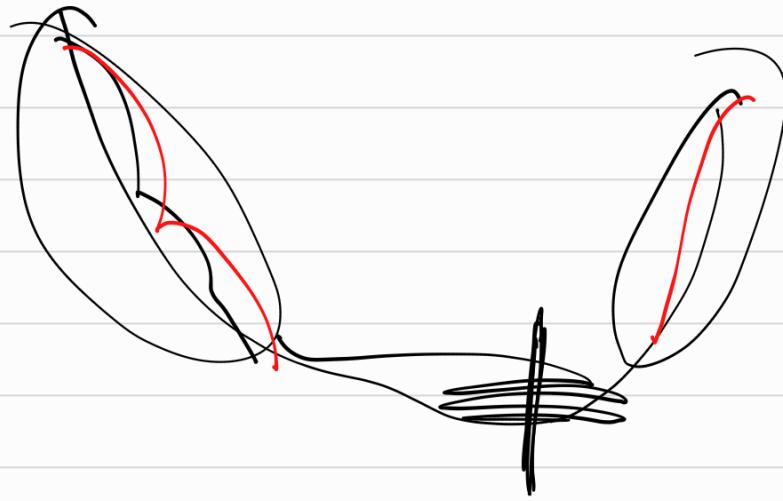


$i = i + \underline{\text{S}}$

25







$a_0=0$

$$a_t = a_{t-1} + s$$

$$a_1 = 0 + s$$

$$a_1 = s$$

$$a_2 = a_1 + s$$

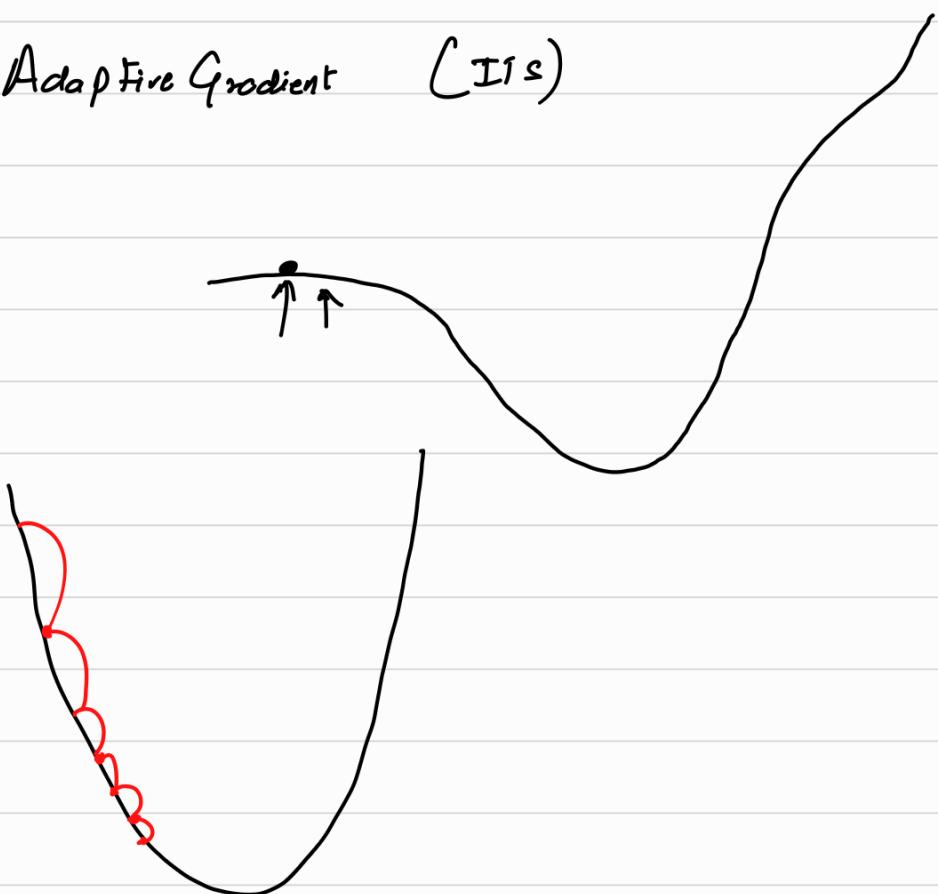
$$a_2 = 10$$

$$a_3 = a_2 + s$$

$$a_3 = (a_1 + s) + s$$

Break!

Adagrad: Adaptive Gradient (It's)



Derivation of the Adagrad update rule:

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{v_{t+1}}} \frac{dh}{d w_t}$$
$$v_t = v_{t-1} + \left(\frac{dh}{d w_t} \right)^2$$

The diagram shows a coordinate system with a horizontal axis and a vertical axis. A black curve represents the function being optimized. A red arrow points from the origin towards the curve, indicating the direction of the gradient. A red line segment connects the origin to a point on the curve, representing the current weight vector. Red arrows on this segment indicate the iterative steps of the Adagrad algorithm as it moves towards the minimum of the function.

Rms Prop

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{v_{t+\epsilon}}} \frac{\partial h}{\partial w_t}$$

v_t $B v_{t-1} + \frac{(1-B)}{\tau} \left(\frac{\partial h}{\partial w_t} \right)^2$

 0.99 0.01

Adam:

Adaptive moments

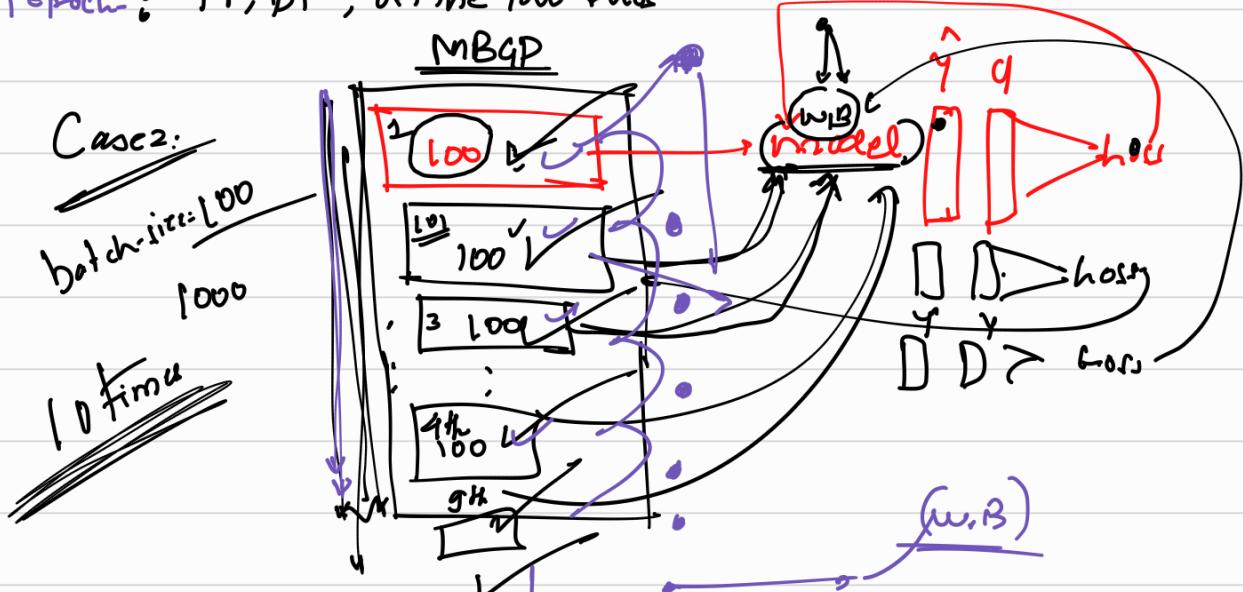
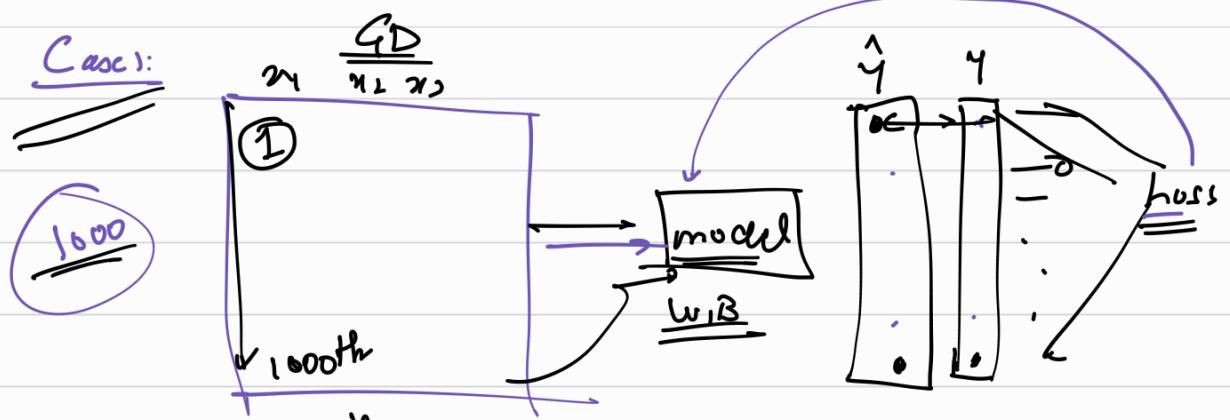
$$\underline{m}_t = \underline{\beta}_1 (\underline{m}_{t-1}) + (1-\underline{\beta}_1) \left(\frac{\partial h}{\partial w_t} \right)$$

$$\underline{v}_t = \underline{\beta}_2 (\underline{v}_{t-1}) + (1-\underline{\beta}_2) \left(\frac{\partial^2 L}{\partial w_t^2} \right)$$

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{v_{t+\epsilon}}} \times \underline{m}_t$$

Mini Batch Gradient Descent \rightarrow Batch Normalization

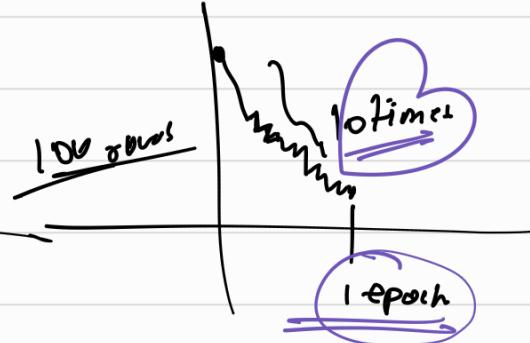
Calculus



Case 1 randomly without replacement



Case 2



model-fit (∞)

model-fit (20)

Batch Normalization ✓
Callbacks ←