Deep learning

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

ML

Reg

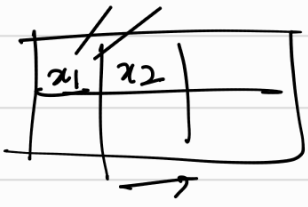
Class.

LR, LR ← Parametric models

DT, RF, Xgboost ← Non-parametric

Process

$y \sim f(x)$

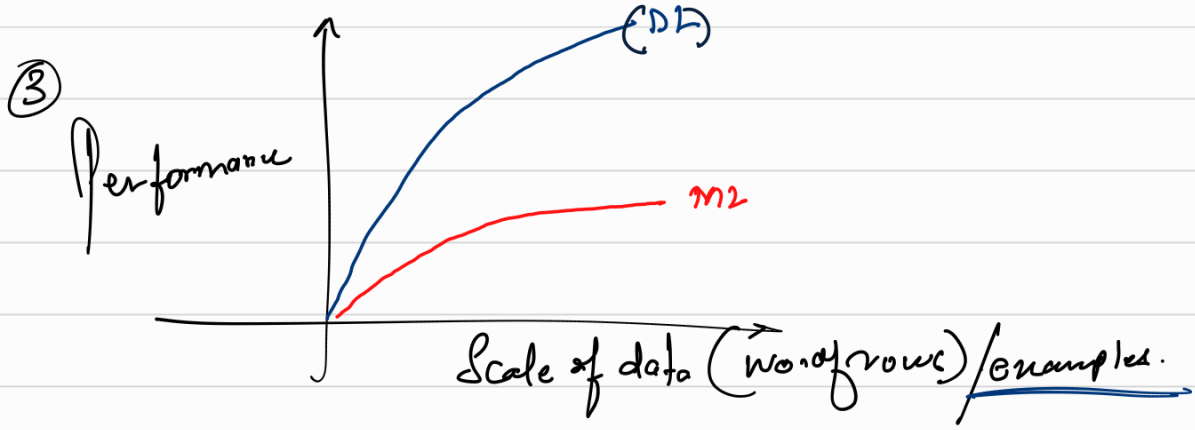


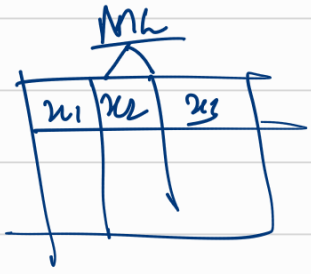
① $x \rightarrow$ (video, audio, text, tabular data, graph).

architecture

multimodality

② Tabular data, text, Scan → model → Diagnosis



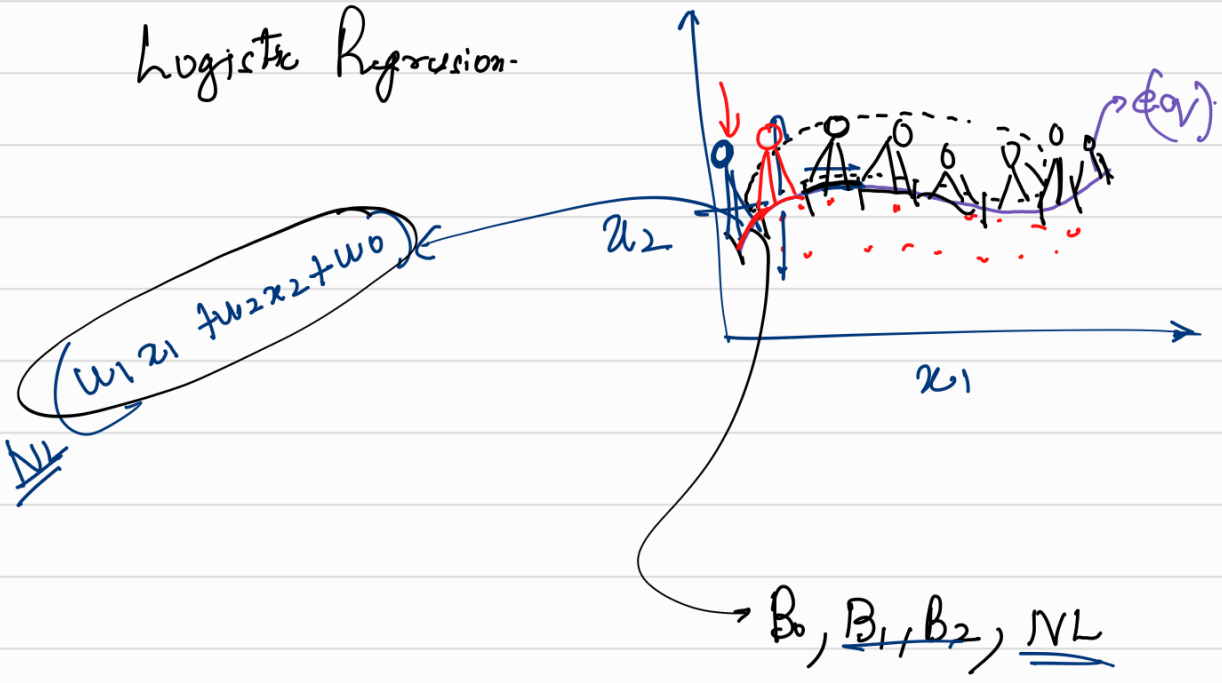


DL → automated feature engineering

DL

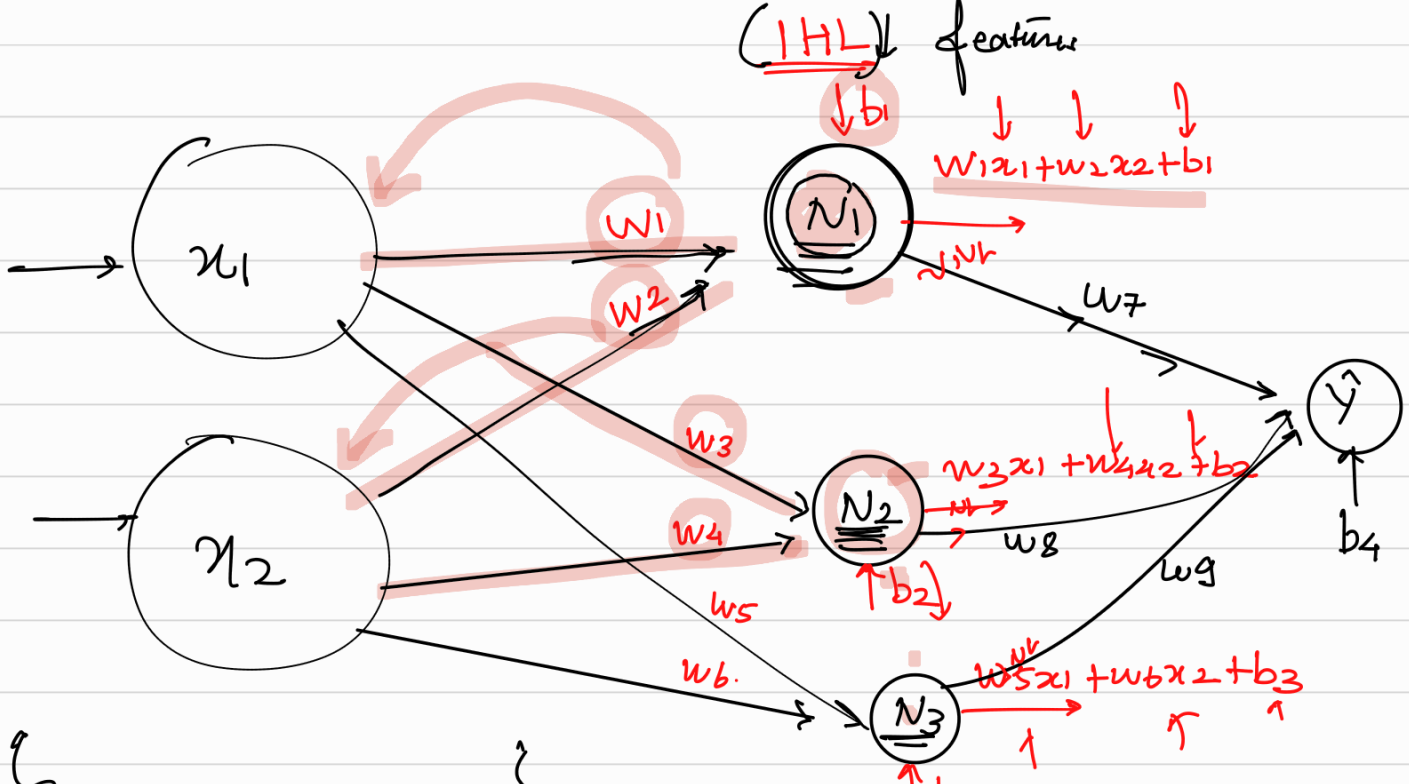
What is a Neuron

Logistic Regression.



B_0, B_1, B_2, NL

x_1	x_2	y



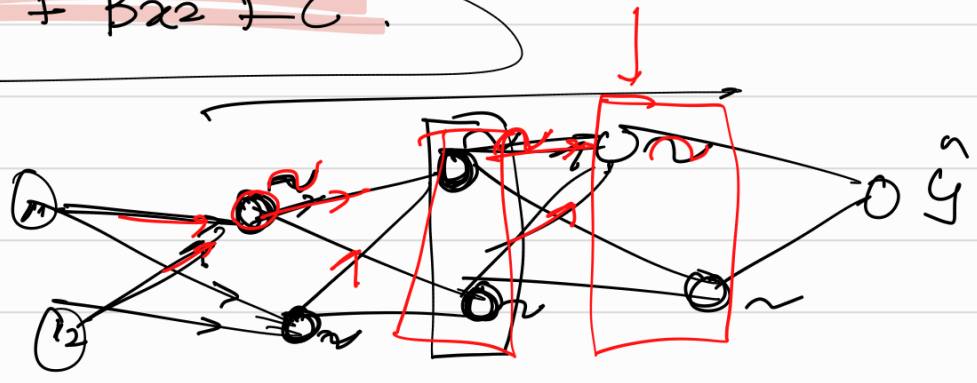
$N_1 = (w_1x_1 + w_2x_2 + b_1)$, $N_2 = (w_3x_1 + w_4x_2 + b_2)$ $N_3 = (w_5x_1 + w_6x_2 + b_3)$

$$\hat{y} = w_7N_1 + w_8N_2 + w_9N_3 + b_4$$

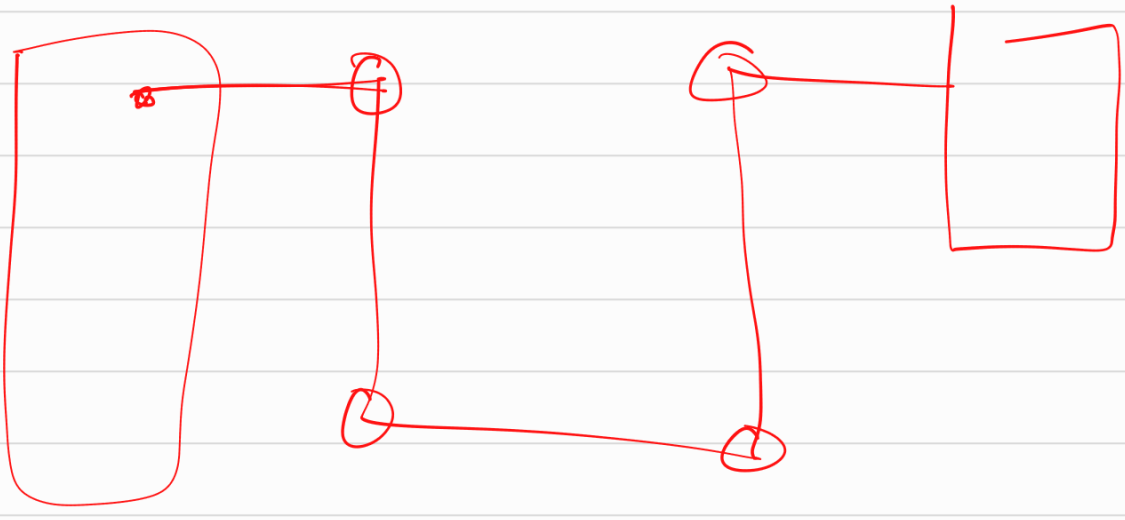
$$\Rightarrow \hat{y} = w_7(w_1x_1 + w_2x_2 + b_1) + w_8(w_3x_1 + w_4x_2 + b_2) + w_9(w_5x_1 + w_6x_2 + b_3) + b_4$$

$$\Rightarrow \hat{y} = x_1(w_7w_1 + w_8w_3 + w_9w_5) + x_2(w_7w_2 + w_8w_4 + w_9w_6) + w_7b_1 + w_8b_2 + w_9b_3 + b_4$$

$$\Rightarrow \hat{y} = Ax_1 + Bx_2 + C$$




1 HL
nth HL

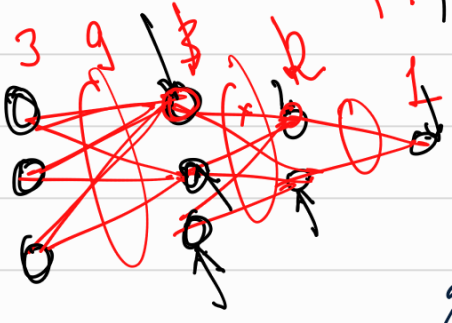


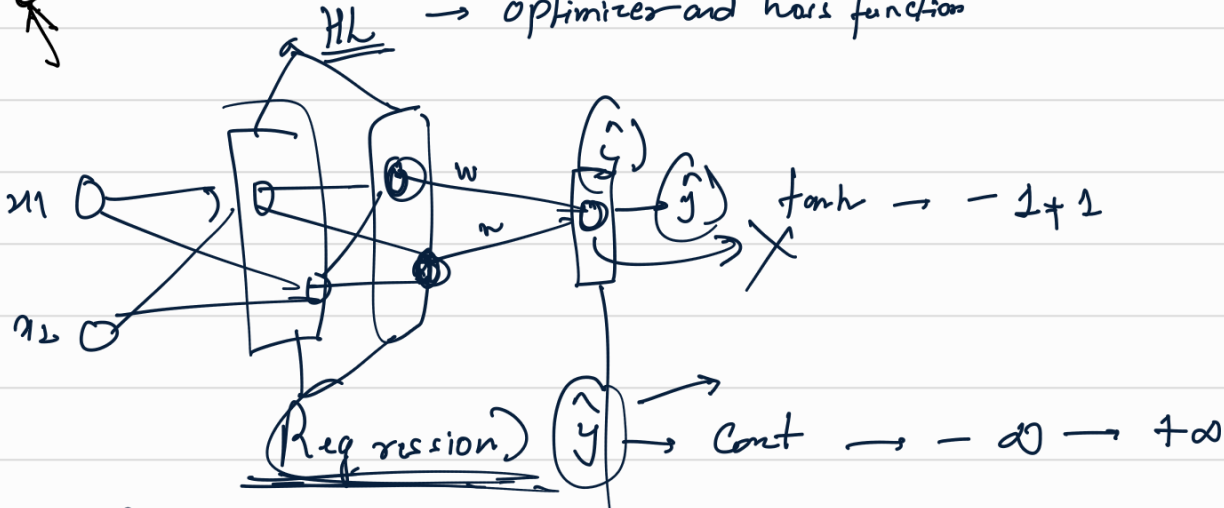
Gradient Descent : minimise the loss by updating the W, B. for all neurons.

Random

1) Define the neural network architecture

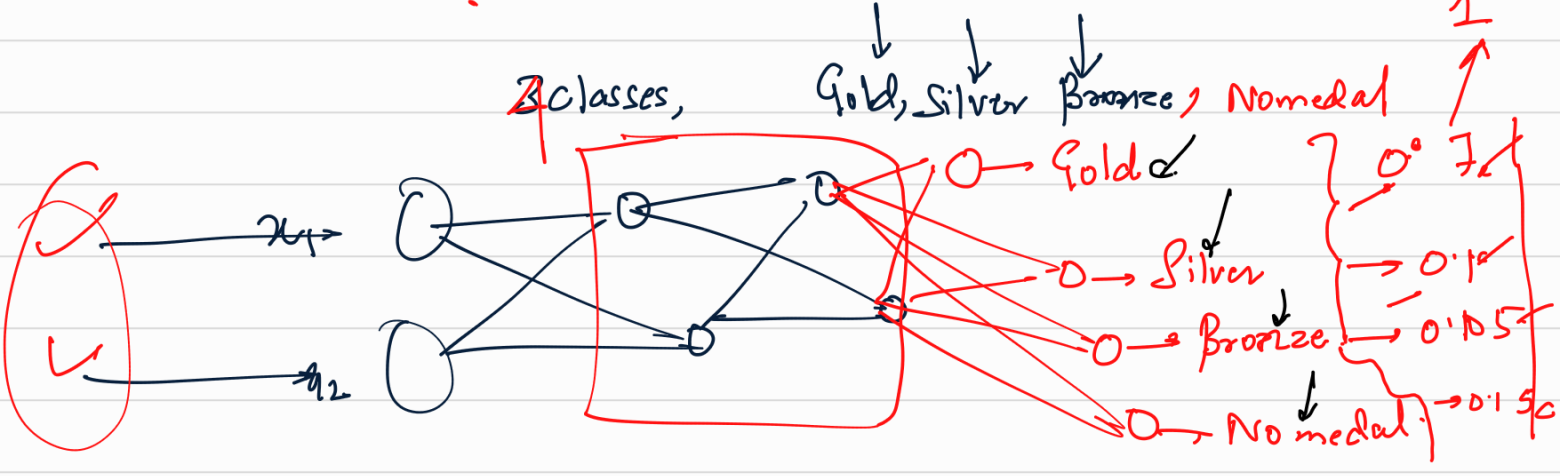
W, b → HLs → Hidden layers
→ Neurons
→ activation function
→ weights and biases → Random
→ optimizer and loss function

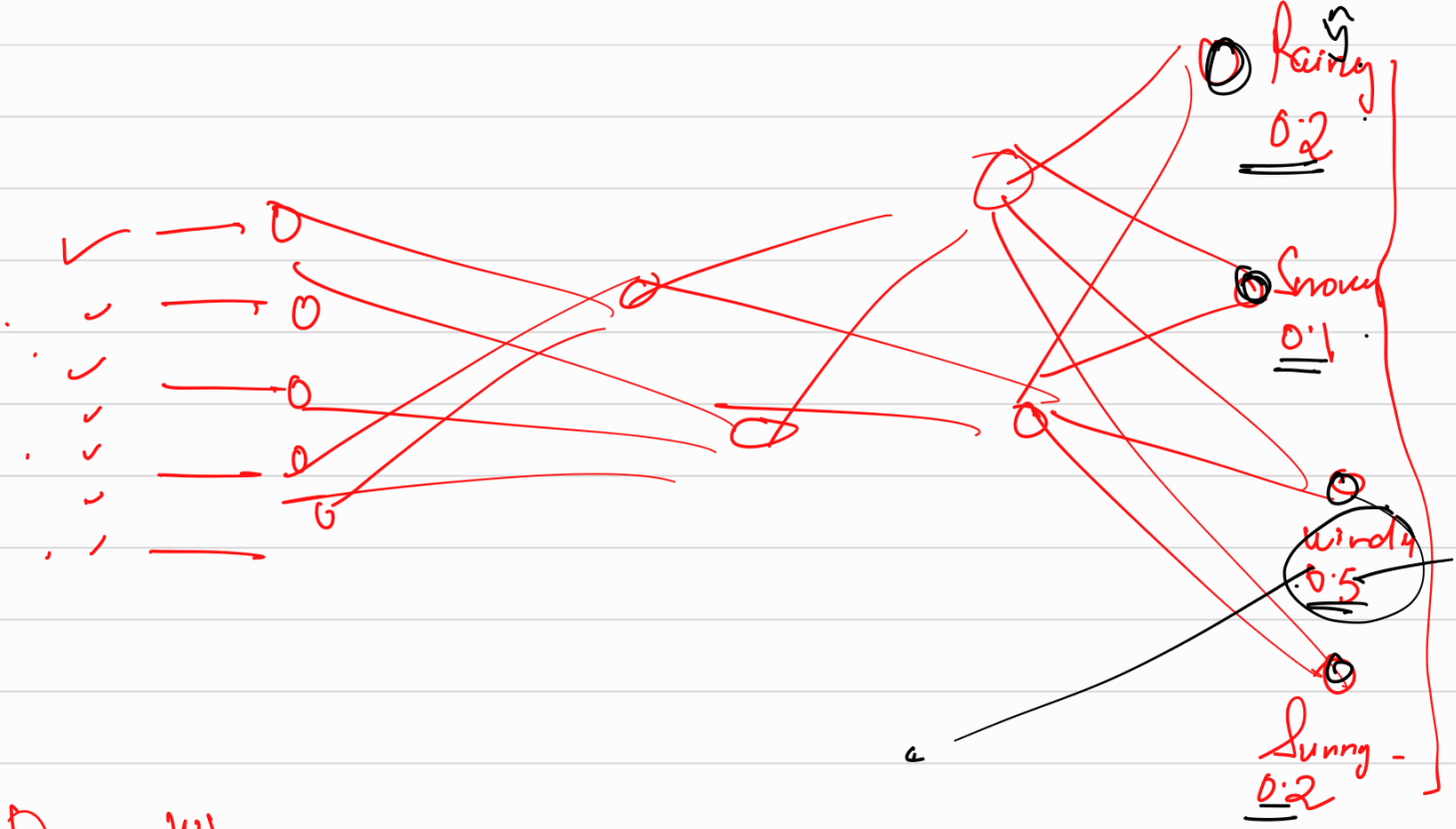




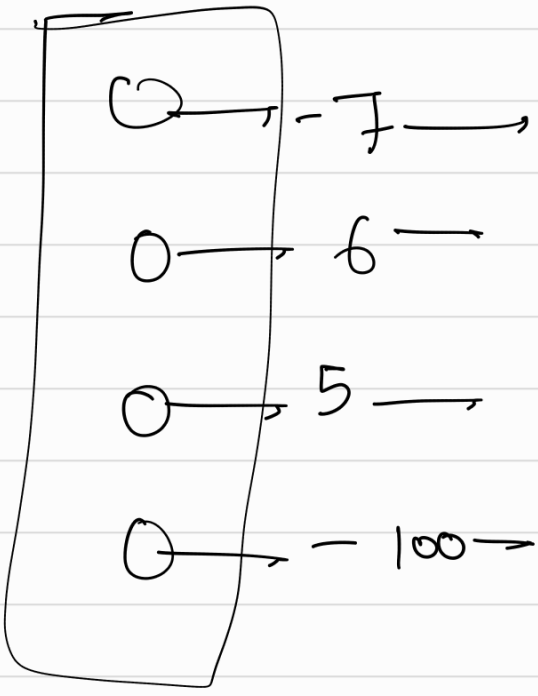
Classification (Binary). → $sig(\hat{y})$ → 0 - 1

Classification (multiclass). Softmax ($\hat{y}$).

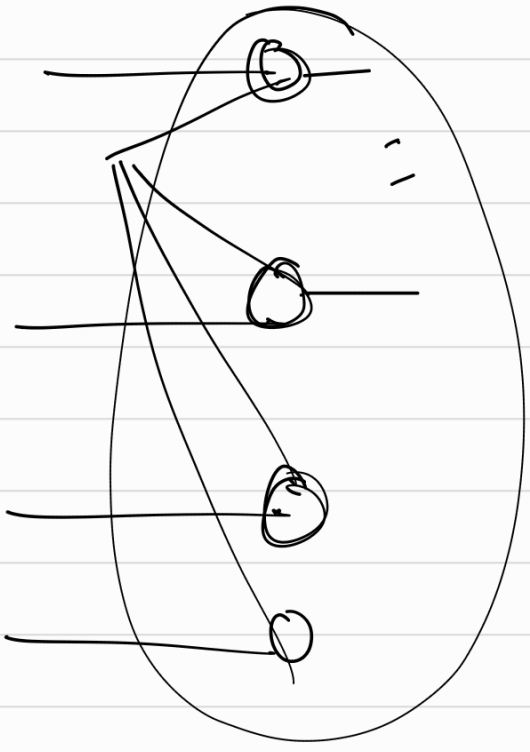


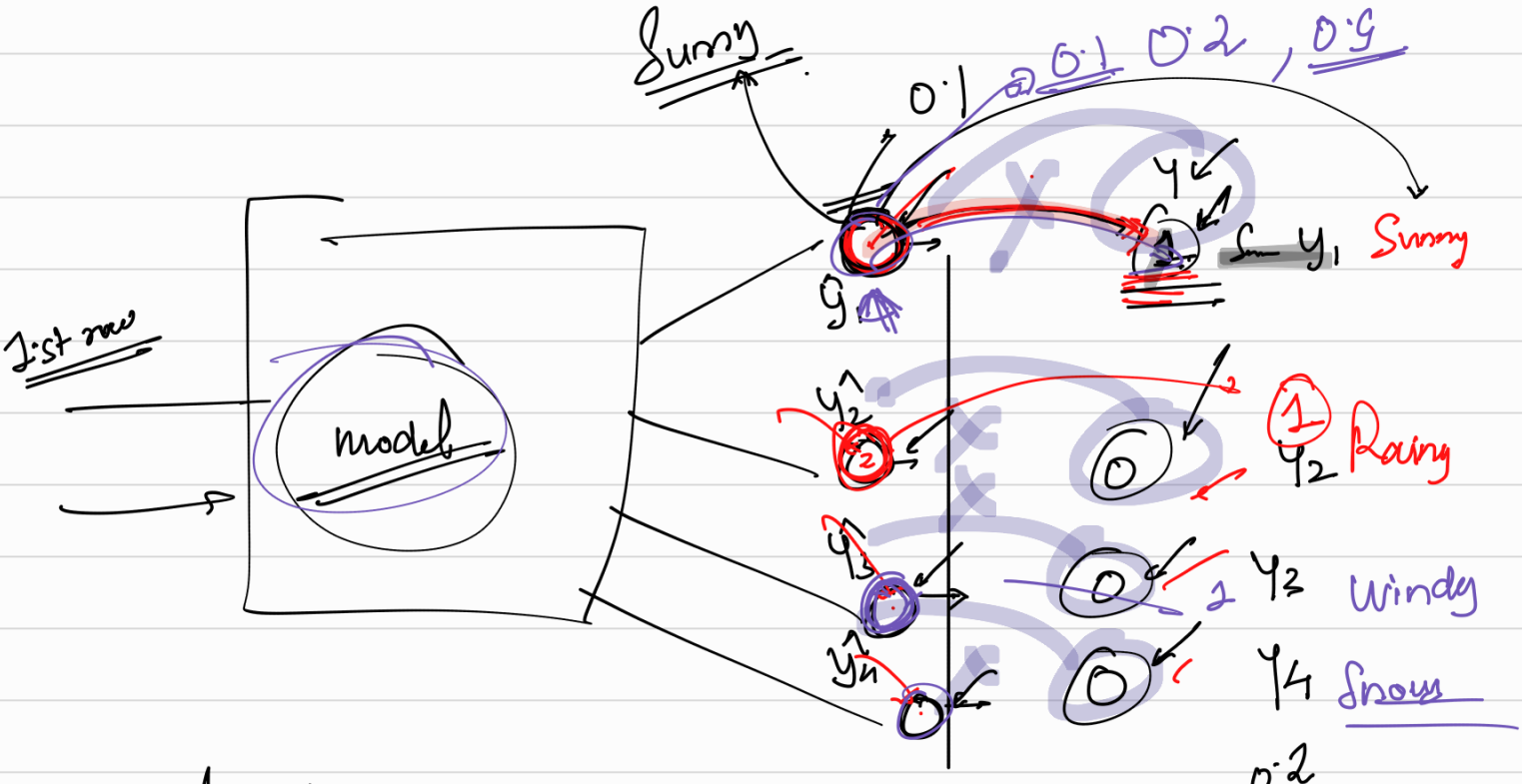

Rainy
0.2
Snowy
0.1
Windy
0.5
Sunny
0.2


x
w1
x
w


7
6
5
100

x_1	x_2	weather (y)	y_Su	y_Ra	y_Wi	y_Sn
		Sunny	1	0	0	0
		Rainy	0	1	0	0
		windy	0	0	1	0
		Sunny	1	0	0	0
		Snowy	0	0	0	1





Log loss = for all rows of data sum $-\left(y_1 \log(\hat{y}_1) + y_2 \log \hat{y}_2 + y_3 \log \hat{y}_3 + y_4 \log \hat{y}_4\right)$.

$\hat{y}_1 = 0.2$, $\hat{y}_2 = 0.8$

$- (\log 0.2)$

$= 0.69$

$- (\log 0.8)$

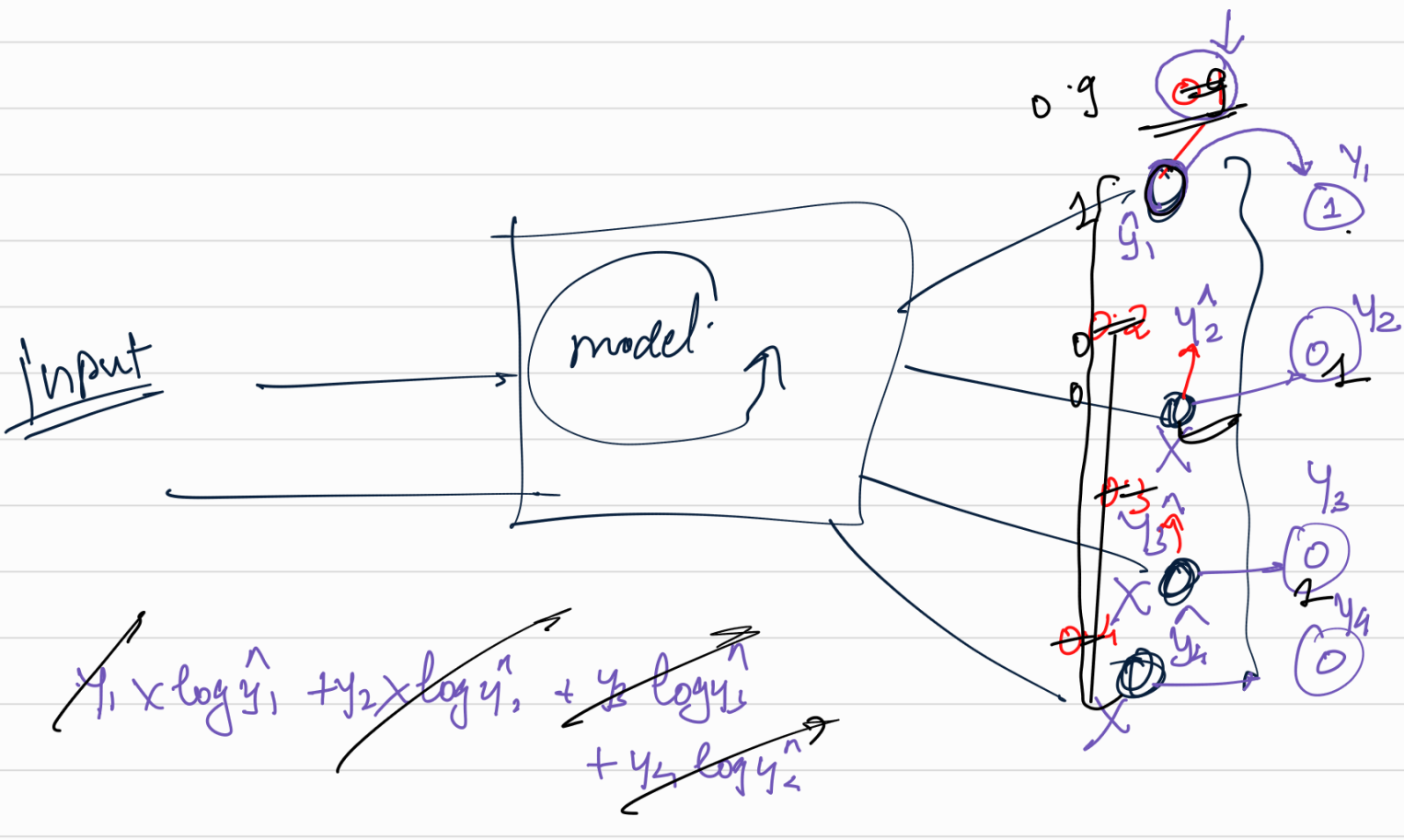
$= 0.09$

$- \log(0.95) = 0.02$

$\log 1 = 0$

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

$$\log(0.1) = \underline{1}$$



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

$\log 1 = 0$

①

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

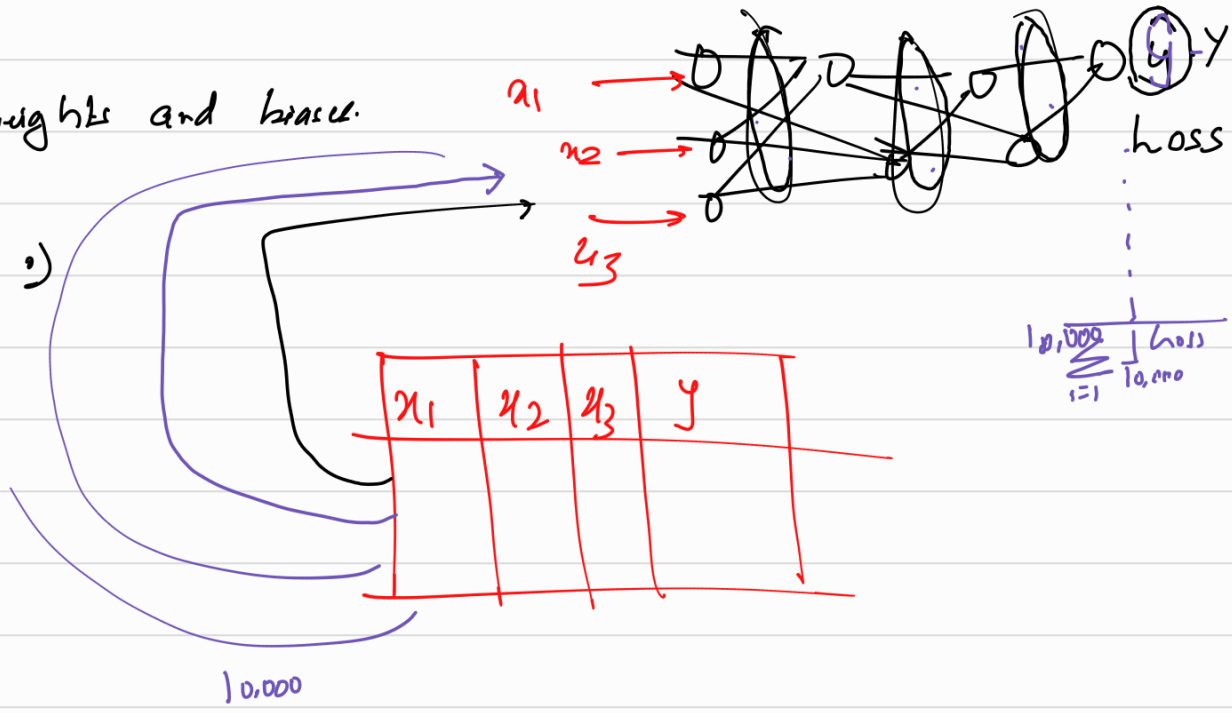
vs

$-\log(0.9) = 0.045$

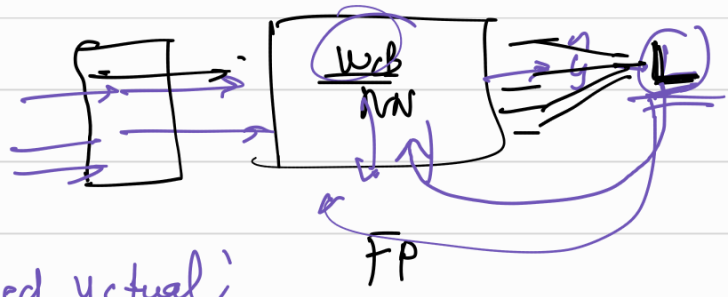
$$\begin{matrix}0\\0\\0\end{matrix} \quad \begin{matrix}0\\0\\0\end{matrix} \rightarrow \text{Softmax, ohe}$$

log Loss

Random weights and biases.



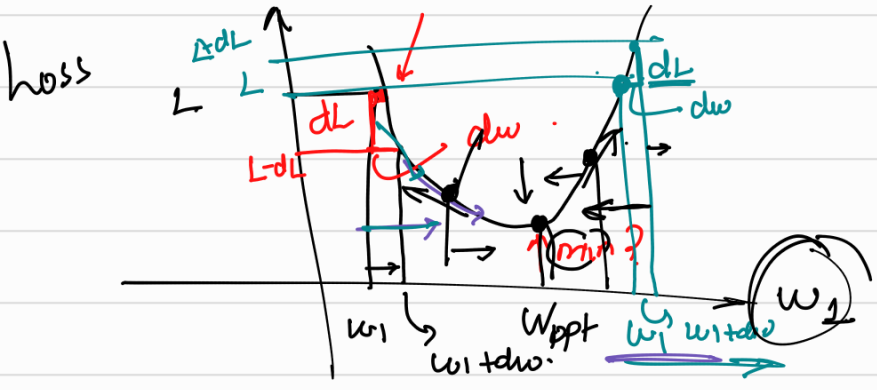
To minimise the loss



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

$L = f((w, B, x), y_{actual})$

$L = f(w, B)$.



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

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

which side walkowd +ve move

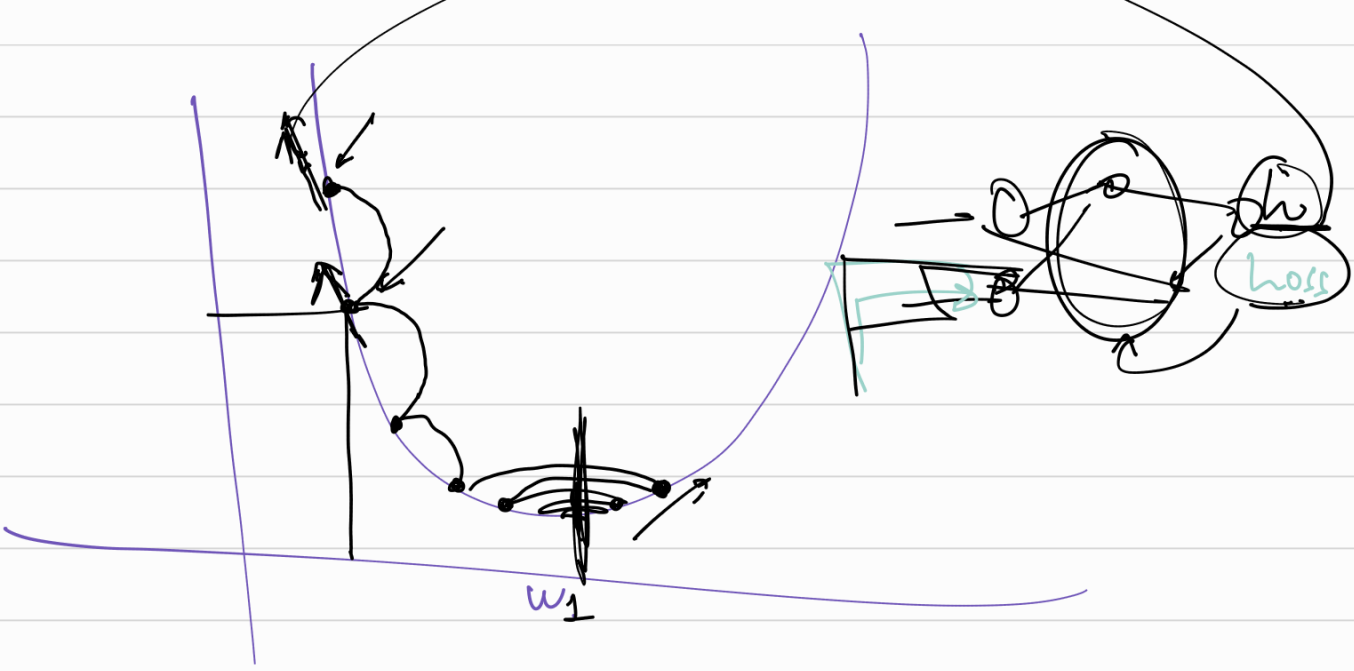
$w = -ve.$

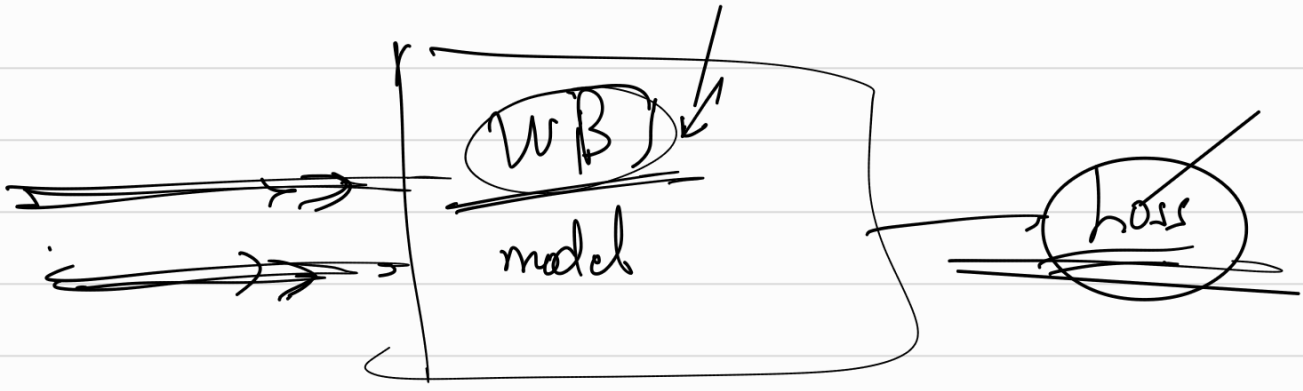
$$w_1(new) = w_1(old) - \alpha\left(\frac{dL}{dw_1}\right)$$

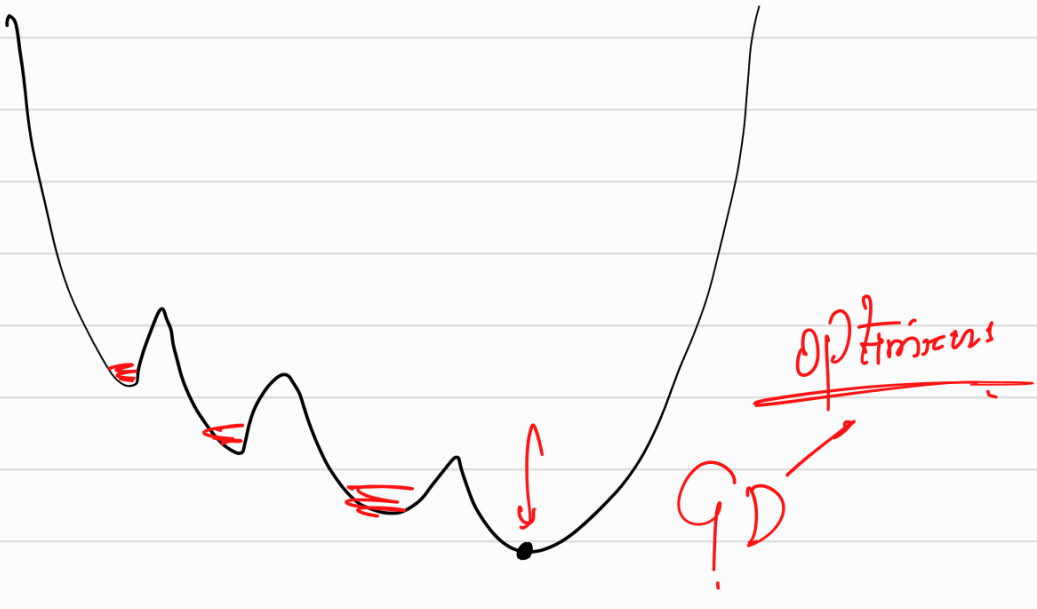
Hyp

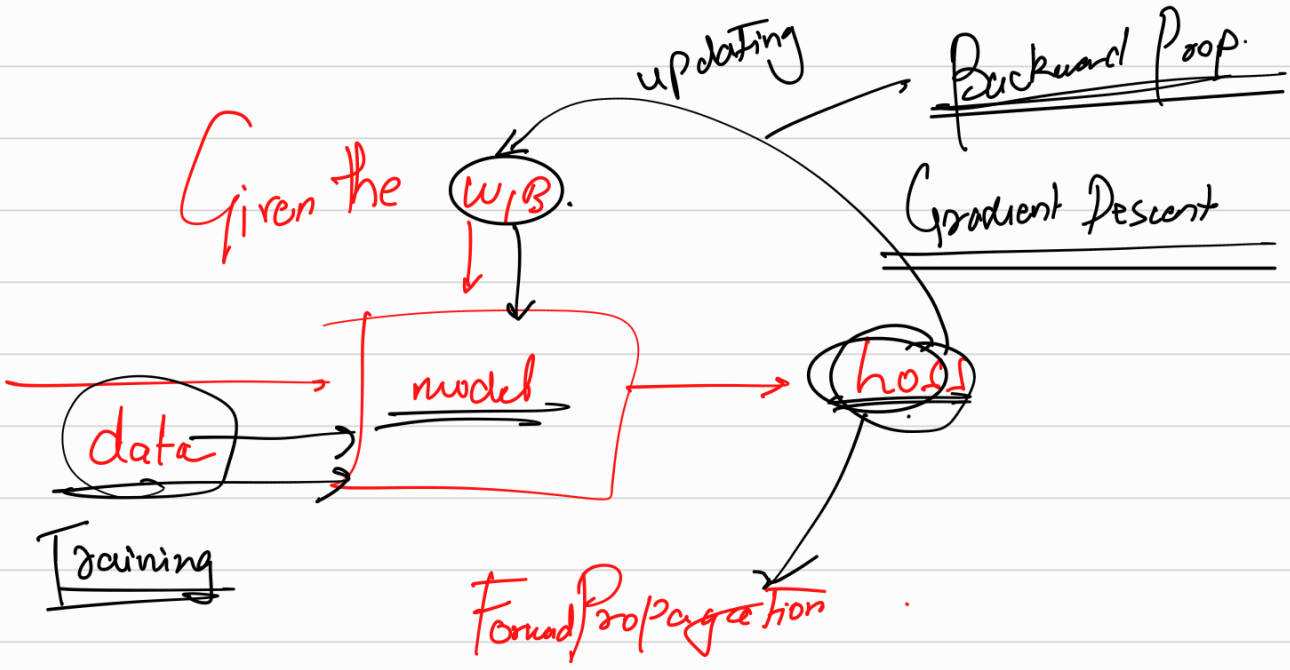
Partial

applicable for all weights and biases.


loss
w_1


WB
model
loss


optimizers
GD



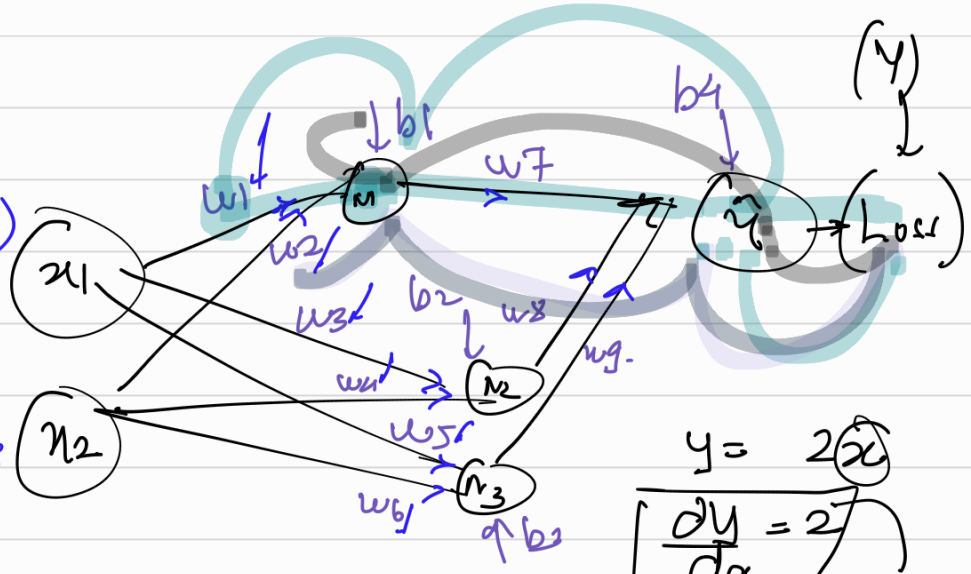
(1FP + 1BP) → epoch

(1FP, 1BP) → 1 epoch.

188 epoch

188 (1FP + 1BP)

lecture-3



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

$N_2 = w_3 x_1 + w_4 x_2 + b_2$

$N_3 = w_5 x_1 + w_6 x_2 + b_3$

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

$y = 2x$

$\frac{dy}{dx} = 2$

$\hat{y} = w_7 N_1$

$\frac{d\hat{y}}{dw_7} = N_1$

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

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

$\frac{\partial L}{\partial w_9}$, $\frac{\partial L}{\partial w_1}$ 6mins

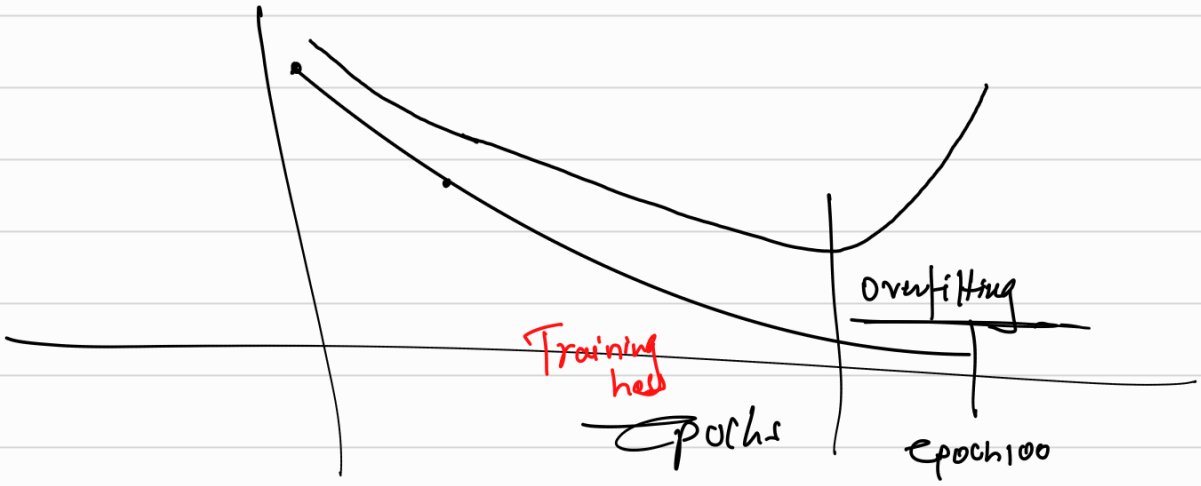
$\frac{\partial L}{\partial w_9} = \frac{\partial L}{\partial \hat{y}} \times \frac{d\hat{y}}{dw_9} \rightarrow N_3$

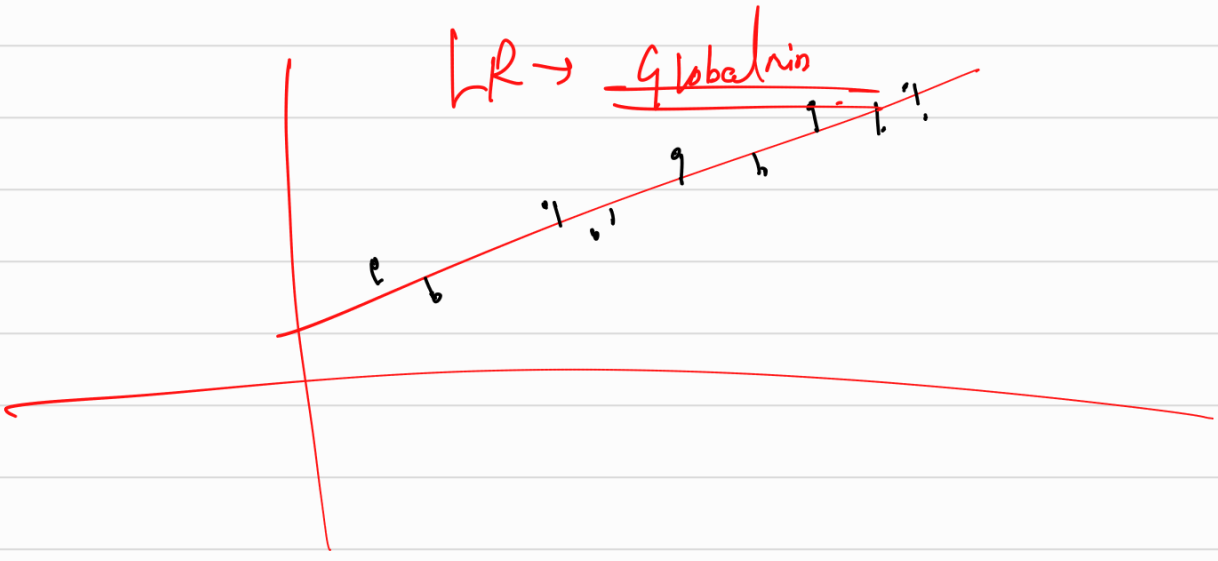
$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{y}} \times \frac{d\hat{y}}{dN_1} \times \frac{dN_1}{dw_1}$

$\hat{y} = w_7 N_1 + w_8 N_2 + w_9 N_3 + b_4$ ($N_2, N_3, b_4 \rightarrow 0$)

$\frac{d\hat{y}}{dw_7} = \frac{\partial(w_7 N_1)}{dw_7} = N_1$

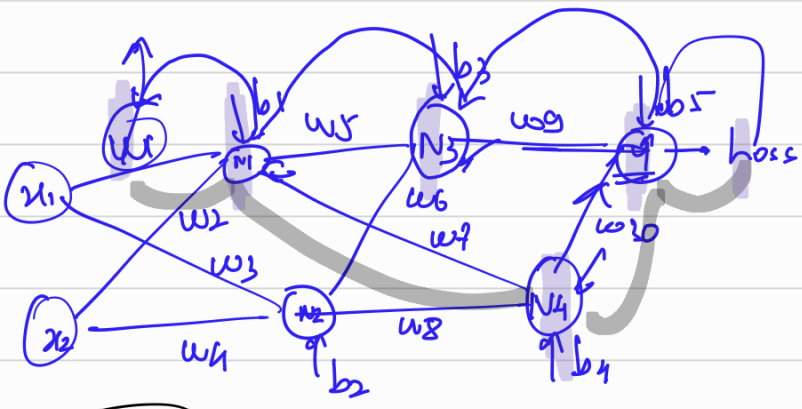
$\frac{d(x \times 2)}{dx} = 2.$


Overfitting
Training
halt
Epochs
epoch100


LR → Global min

Activation functions

$w_1 x_1 + w_2 x_2 + b_1$



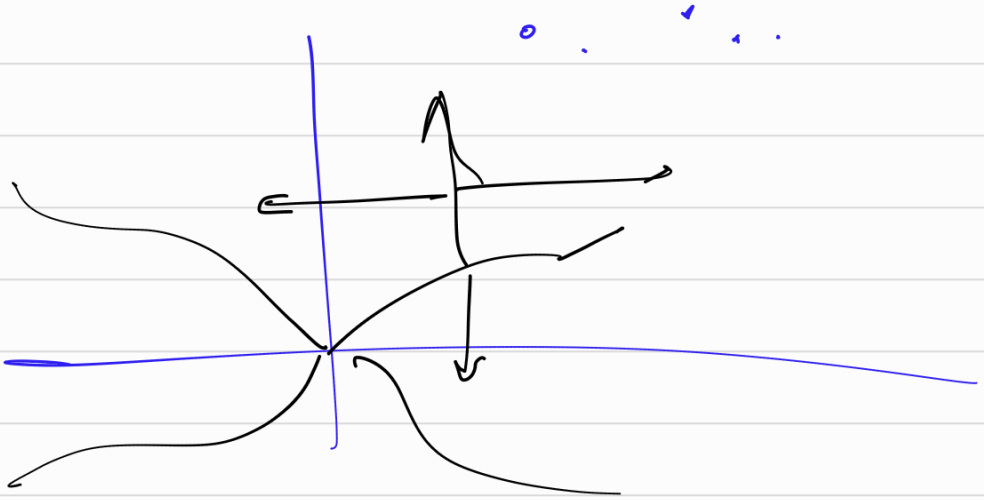
$\frac{\partial L}{\partial w_1}$

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

$\frac{\partial h}{\partial w} = 100000$ Terms $= N(HL) + 2$

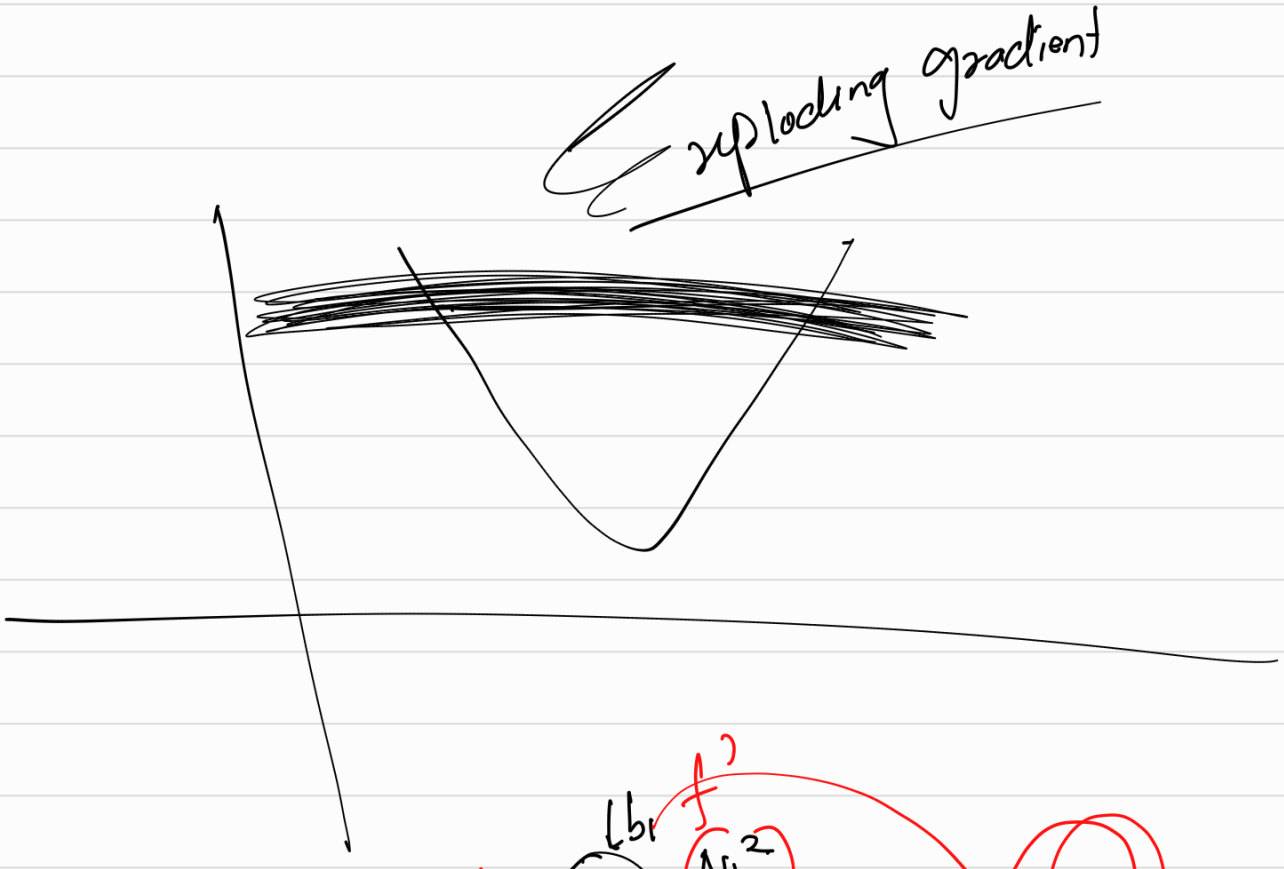
$\frac{\partial h}{\partial w_1} = 0.000002$

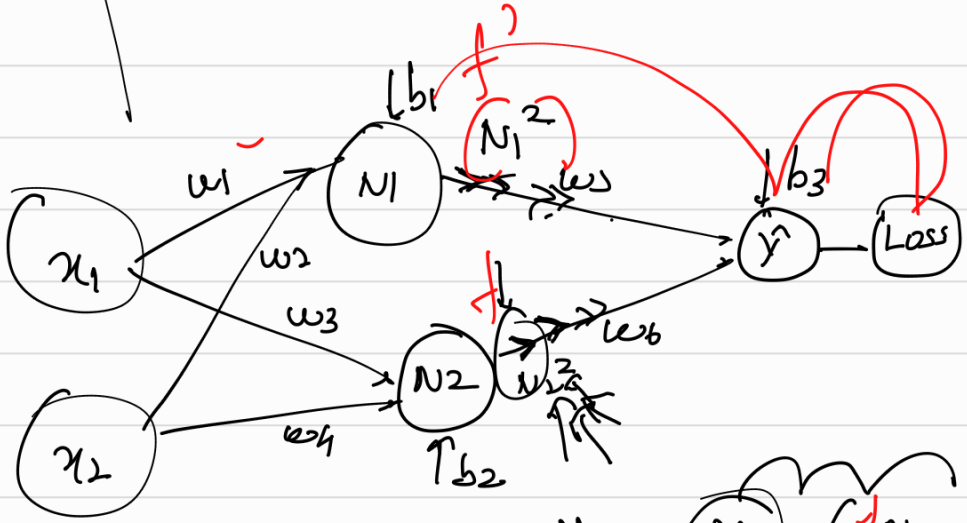
$w_1 -$



$(0.99)^{365} \approx$ 0.03

$(1.01)^{365} =$ 37.78





$N_1 = w_1 x_1 + w_2 x_2 + b_1$

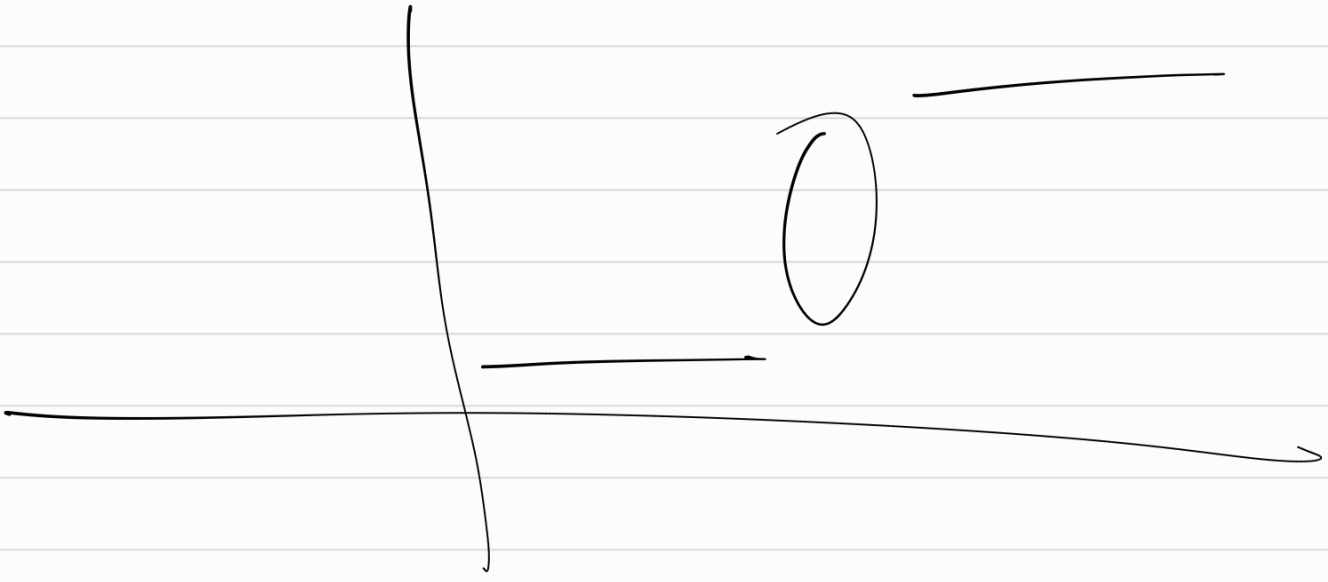
$N_2 = w_3 x_1 + w_4 x_2 + b_2$

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

$$\hat{y} = w_5 (N_1)^2 + w_6 N_2^2 + b_3$$

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

Activation functions → Should be cont & diff

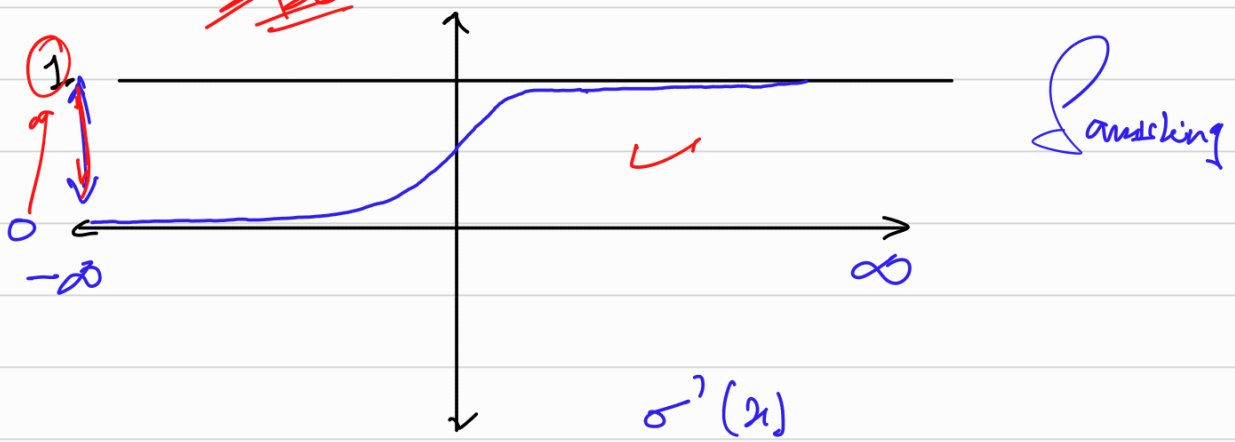


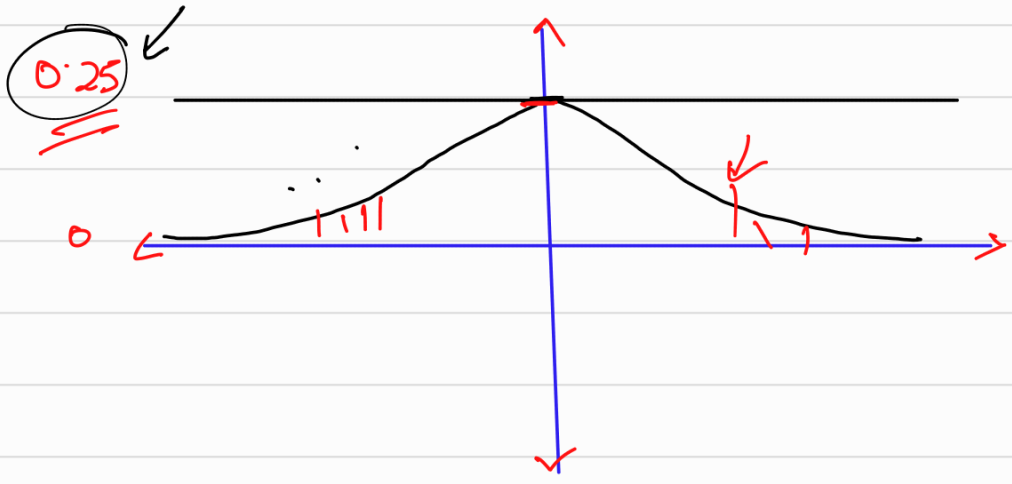
Activation functions are functions which should be continous & diff that can be applied at the output of any hidden layers to create non-linear feature or at the o/p to create bounded outputs → Probs

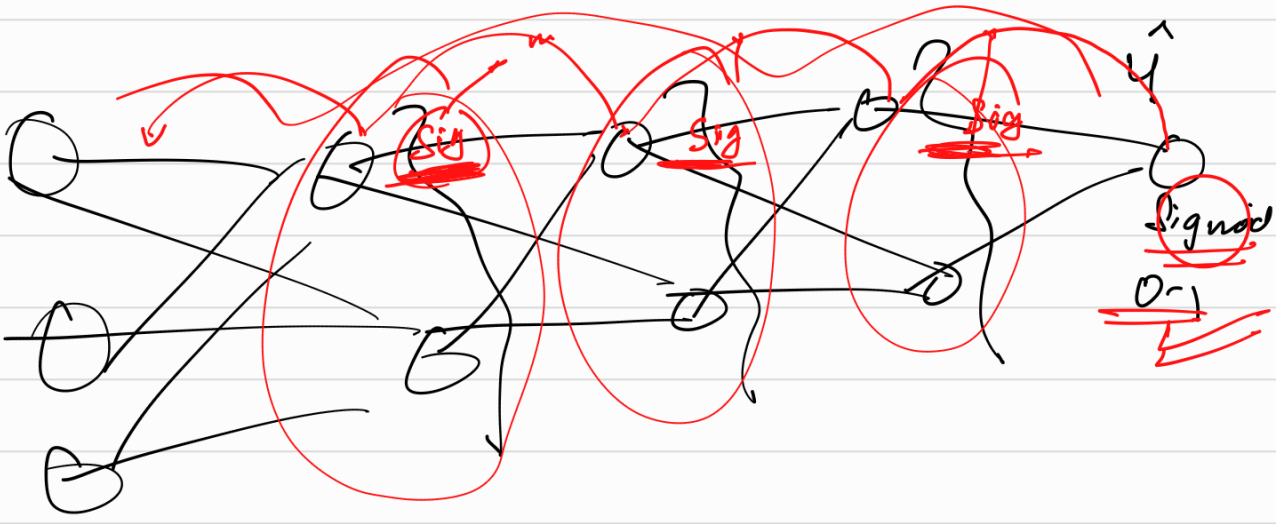
↳ output layer

→ Sigmoid $\sigma(x) = \frac{1}{1+e^{-x}}$ $-\infty < x < \infty$

HLX 0/L Probs $0 < \sigma(x) < 1$

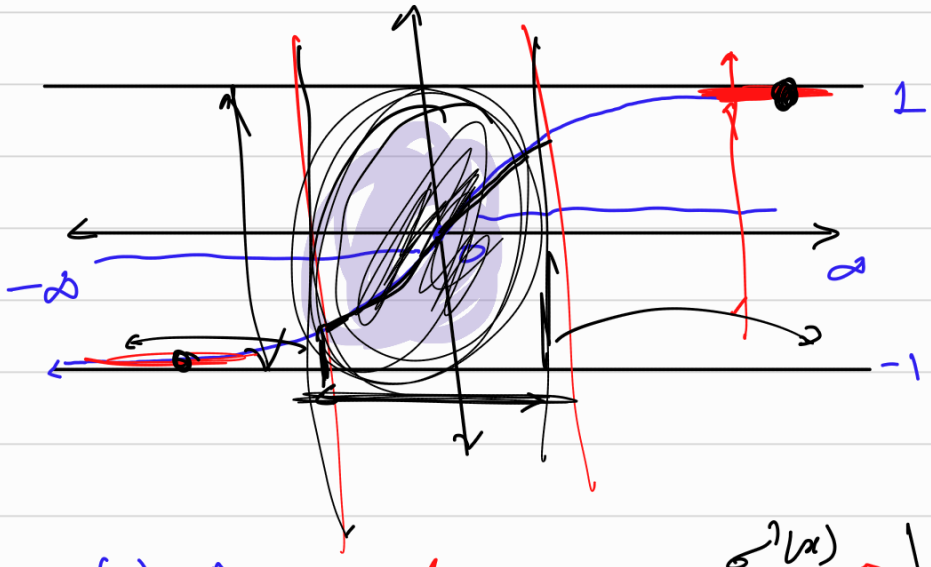


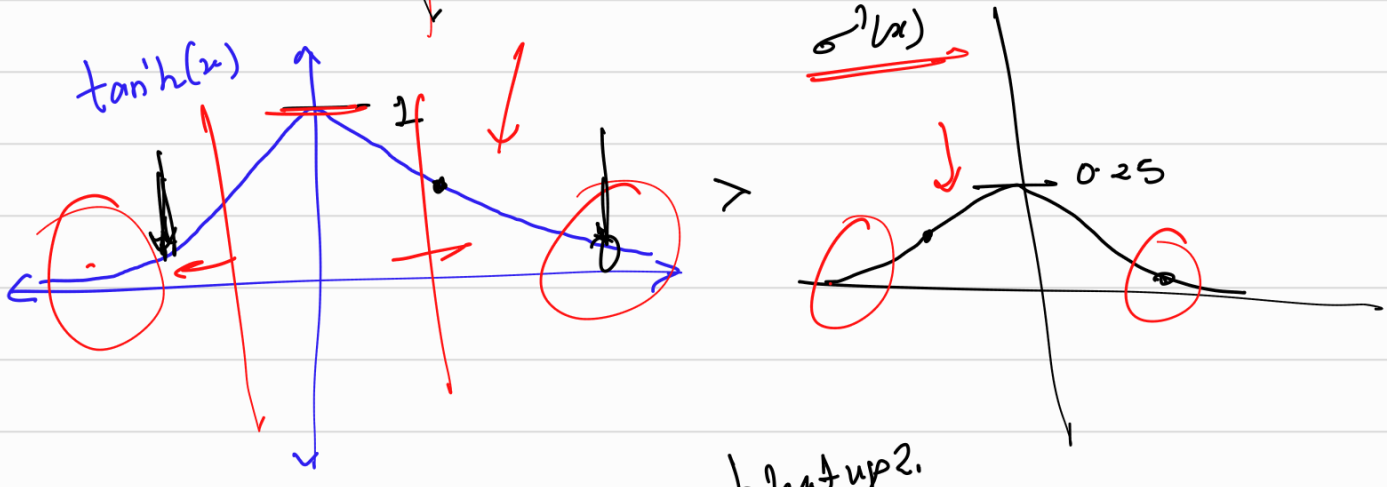


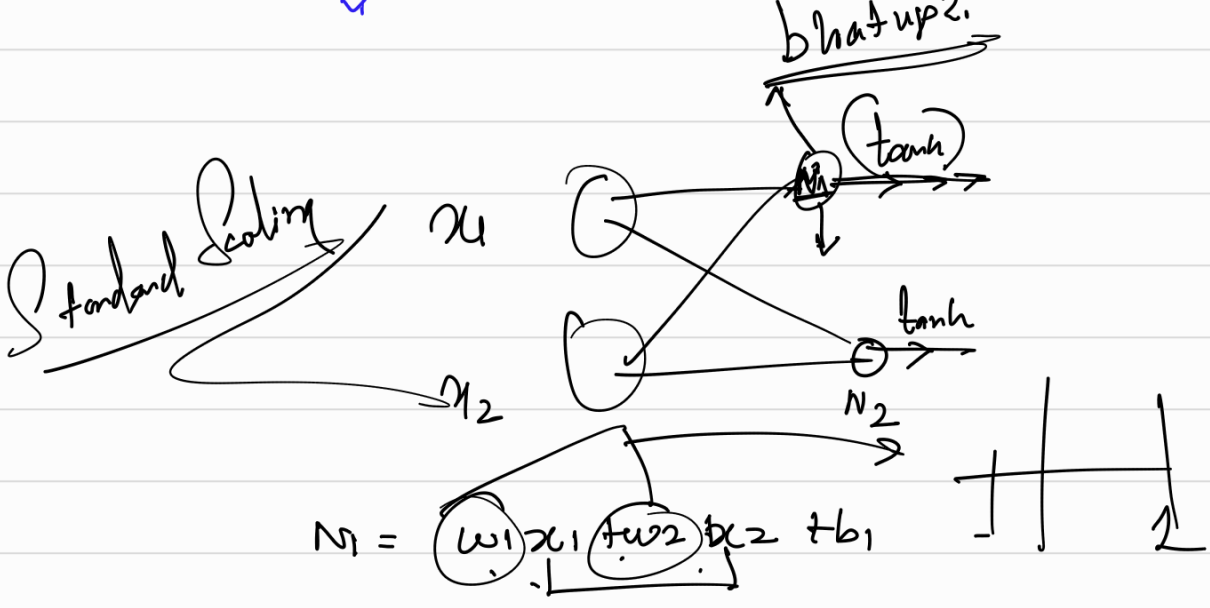


tanh → hyperbolic tangent

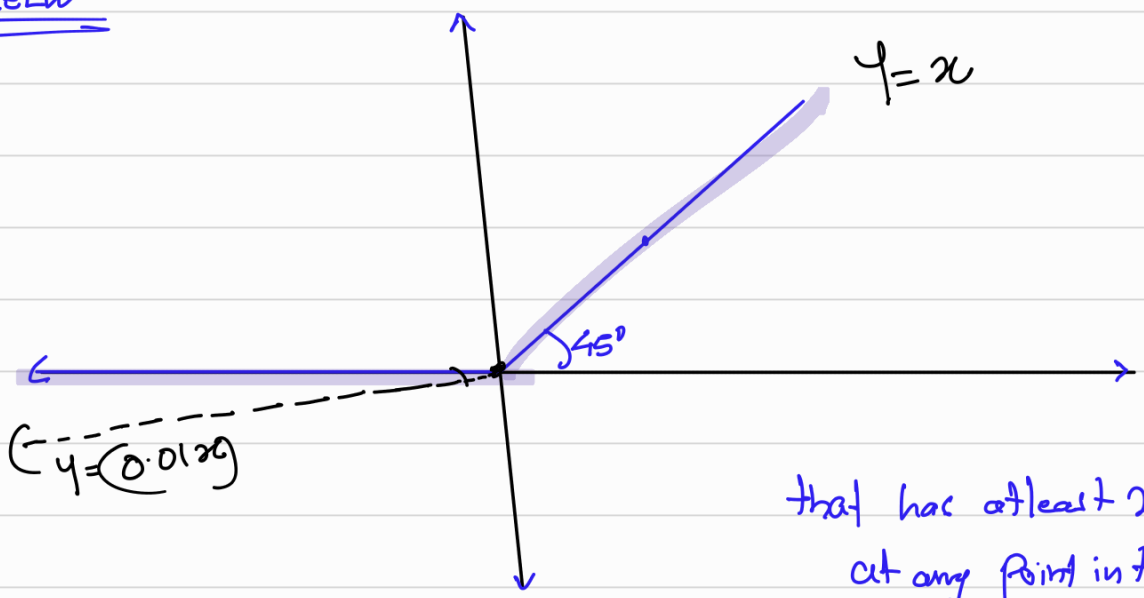
$$\tanh = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$







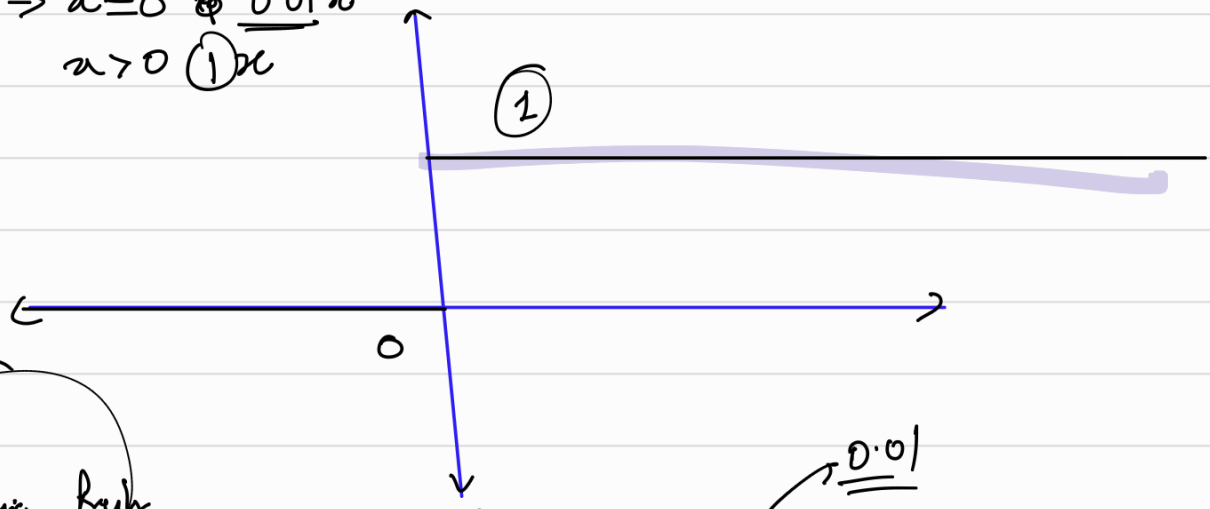
ReLu



that has atleast 2 slopes at any point in the x space.

$$\text{ReLu}(x) = \begin{cases} 0 & x \le 0 \\ x & x > 0 \end{cases}$$

Leaky ReL(x) $\Rightarrow$ $x \le 0$ $0.01x$; $x > 0$ (1) x

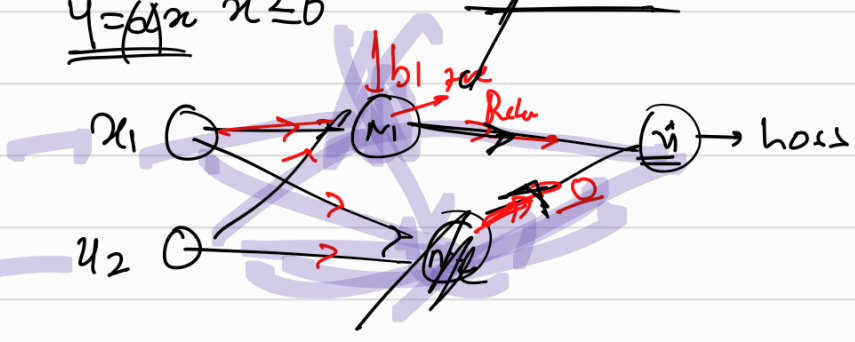


Parametric ReLu

$y = x$ $x > 0$

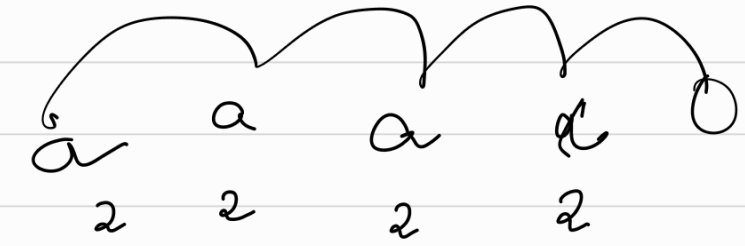
$y = \alpha x$ $x \le 0$

$0 < \alpha < 1$ — 0.01



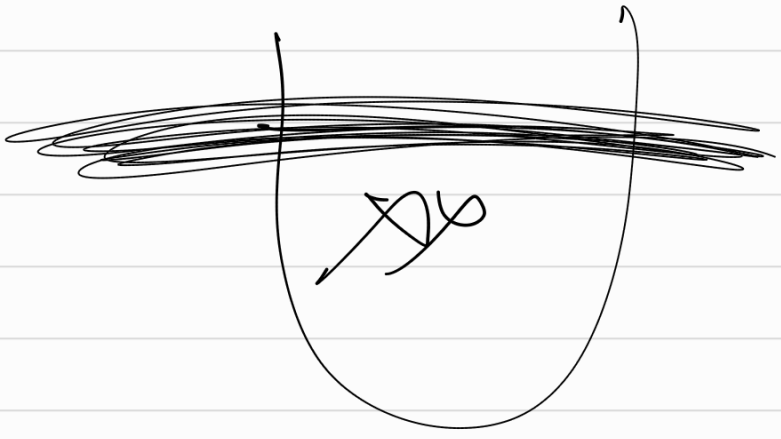
Deep Neural network

$(2)^{0+10+50} = 32$

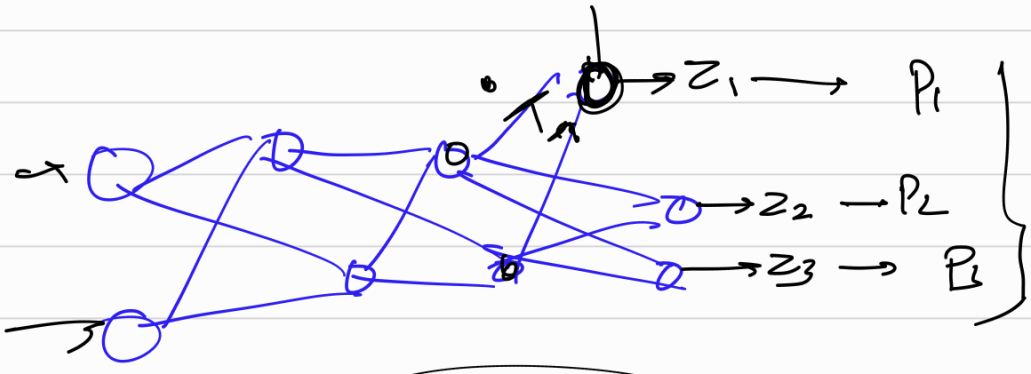


(2) $(2)^{100}$

$$\frac{dw}{dw_1} = w_1 - \alpha \left(\right)$$

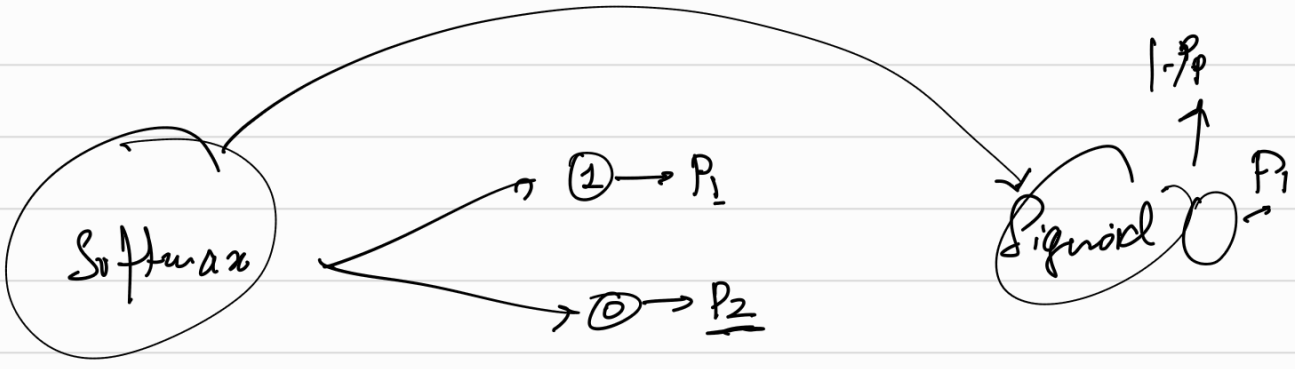


Softmax : needs to be applied at the o/p of a multiclass problem



$$p_1 + p_2 + p_3 = 1$$

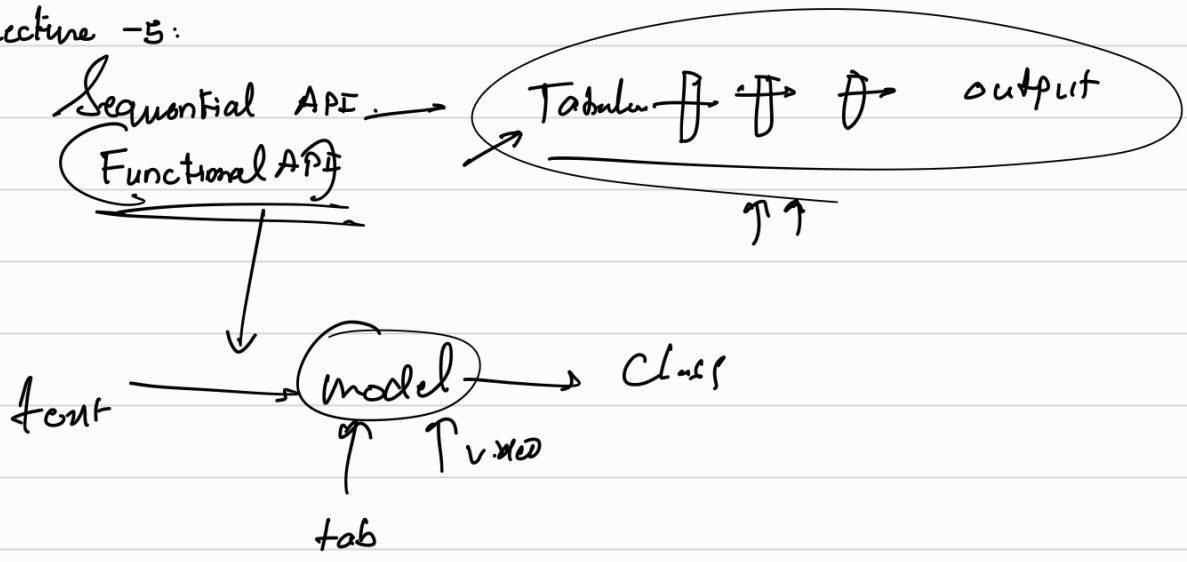
$$p_1 = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}} \qquad p_2 = \frac{e^{z_2}}{e^{z_1} + e^{z_2} + e^{z_3}} \qquad p_3 = \frac{e^{z_3}}{e^{z_1} + e^{z_2} + e^{z_3}}$$



$$p_1 + p_2 = 1$$
$$p_1 = 1 - p_2$$
$$p_2 = 1 - p_1$$

Break

Lecture -5:



Sparse.

$y \rightarrow$ LE (0, 1, 2)

$(y = 1)$

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

0 1 2

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

$-\log 0.3$

$y \rightarrow$ OHE.

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

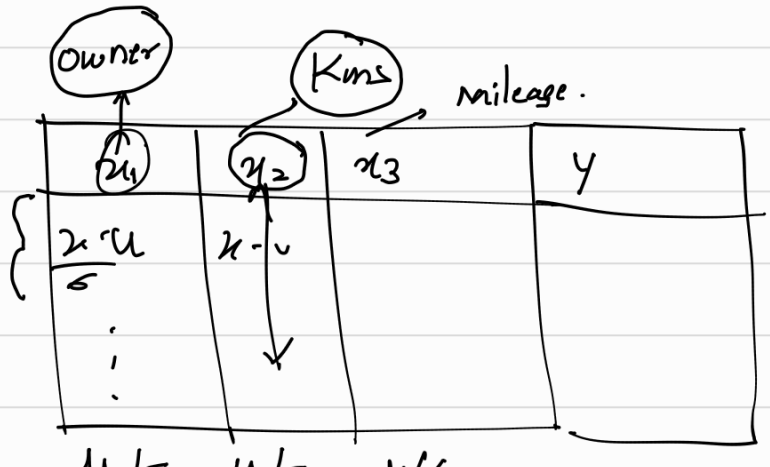
cce.

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

$y = [0 \quad 1 \quad 0]$.

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

$= -\log 0.3$



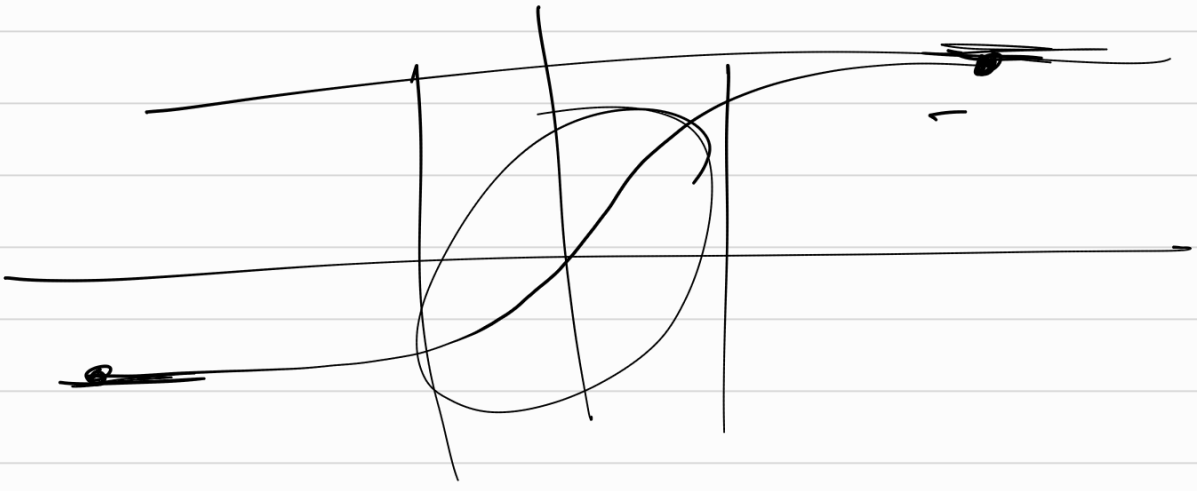
$w_1 \times (\text{owner}) + w_2 \times (\text{Kms}) + w_3 \times \text{mileage}$

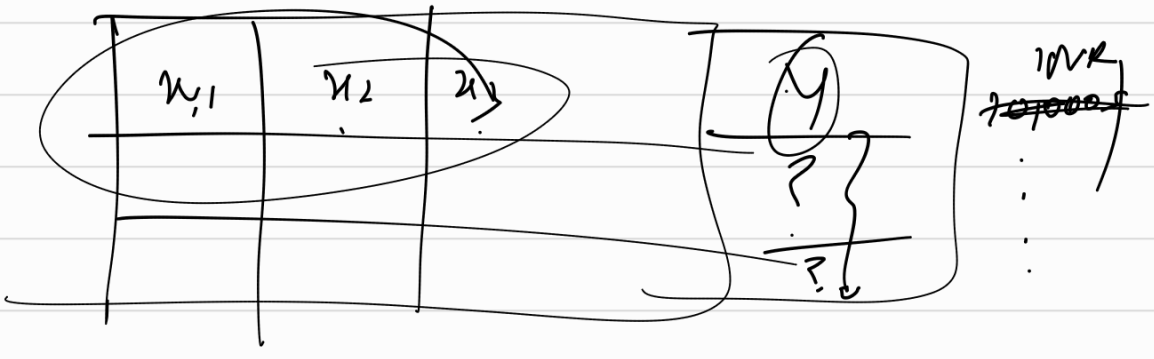
(1) Honda City

50,000 ↑

(2) ~~2nd~~ Honda city

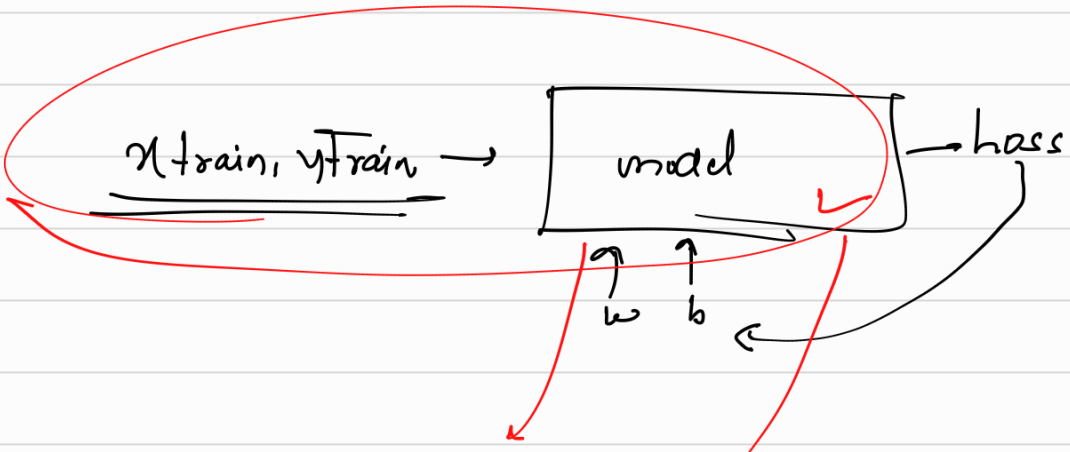
40,000

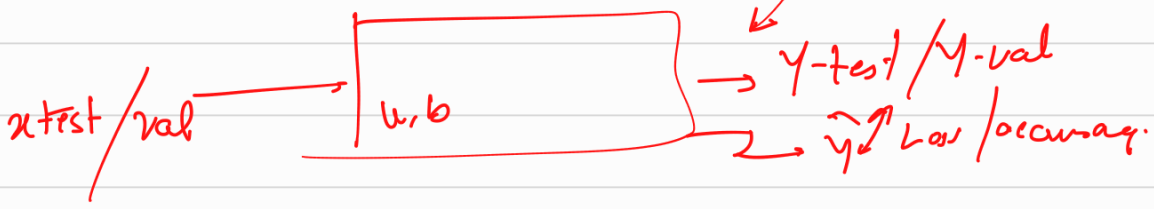


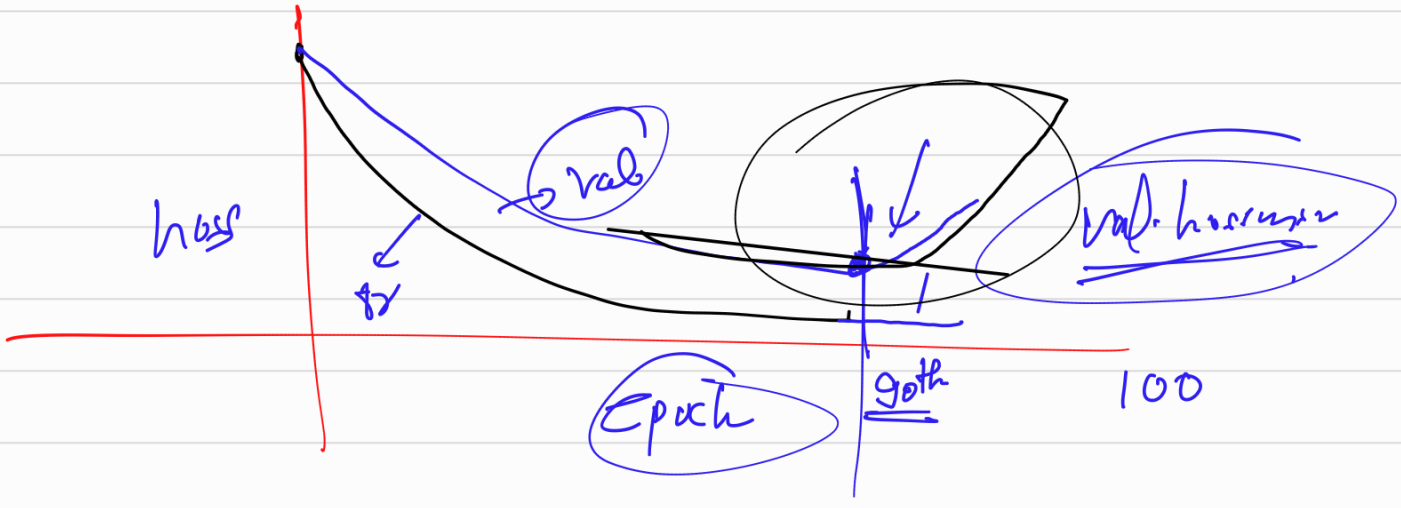


3.3

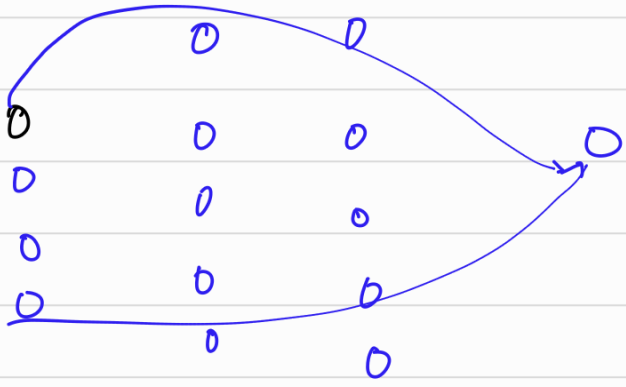
$(w_i \sim -1, 1)$


Xtrain, ytrain
model
loss
w
b


xtest/val
w, b
y-test/y-val
ŷ
Loss/accuracy


loss
val
val. loss
Epoch
90th
100

Callbacks



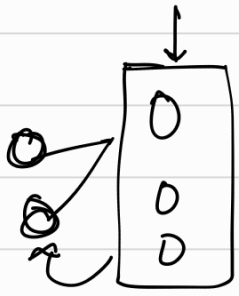
model.fit(train, val = val).

model.evaluate(test)

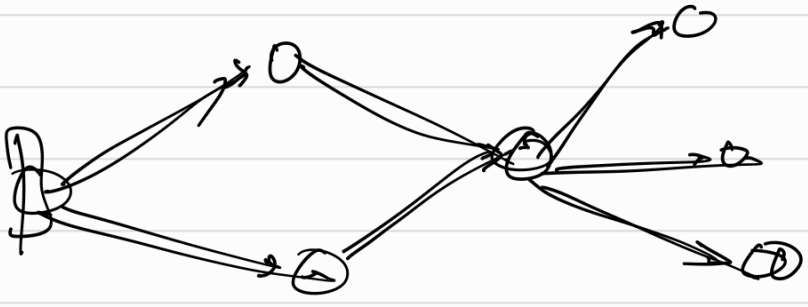
82%

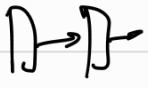
Functional API:

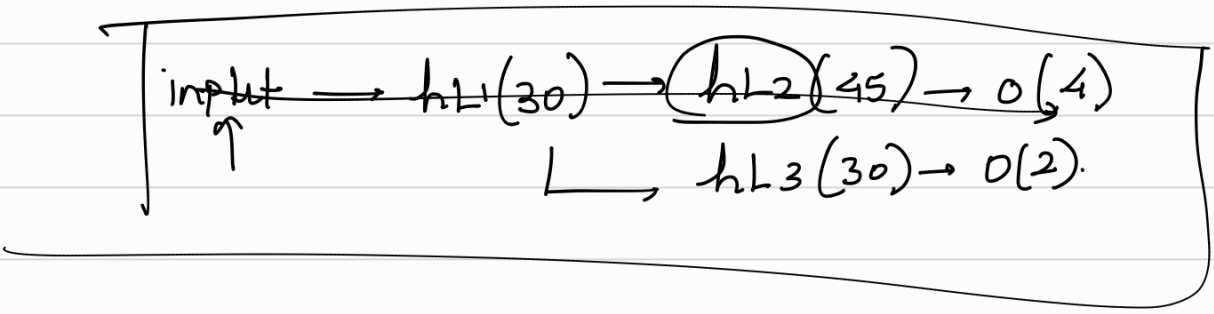
Graph Structure

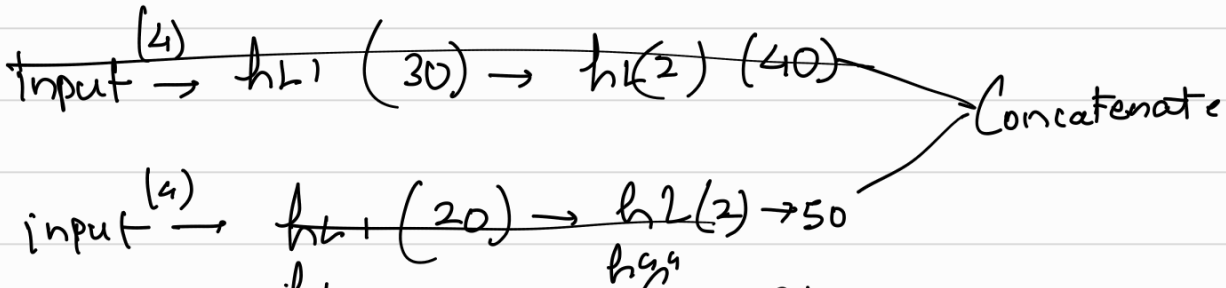


any layer (✓)









Case 1:

inputs = Input (4,)

hL1 = Dense (30) (inputs)

hL2 = Dense (45, activation = 'relu') (hL1)

O1 = Dense (4) (hL2).

hL3 = Dense (30) (hL1)

O2 = Dense (2) (hL3).

Case 2:

inp1 = Input (4,)

inp2 = Input (4,).

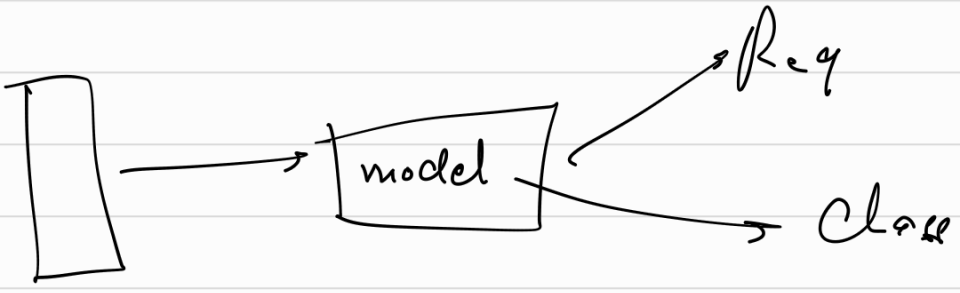
hL1 = Dense (30) (inp1).

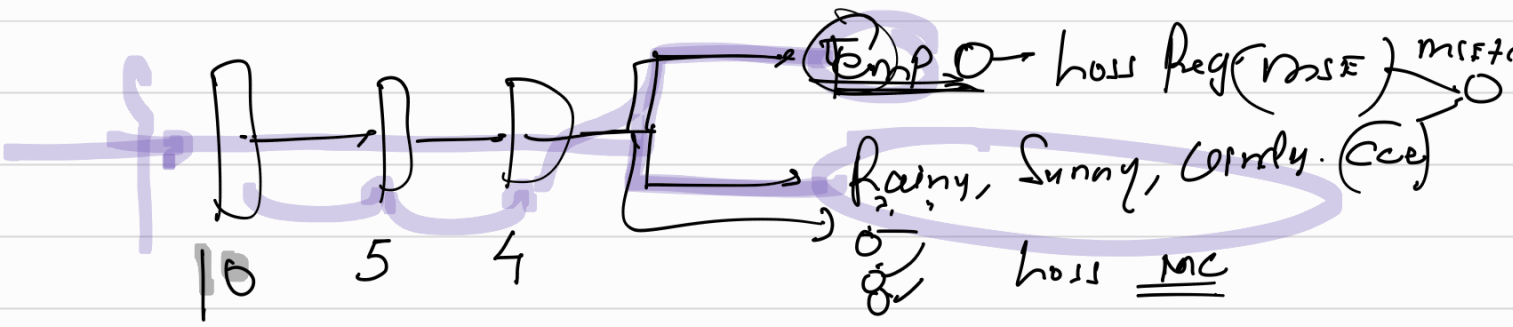
hL2 = Dense (40) (hL1).

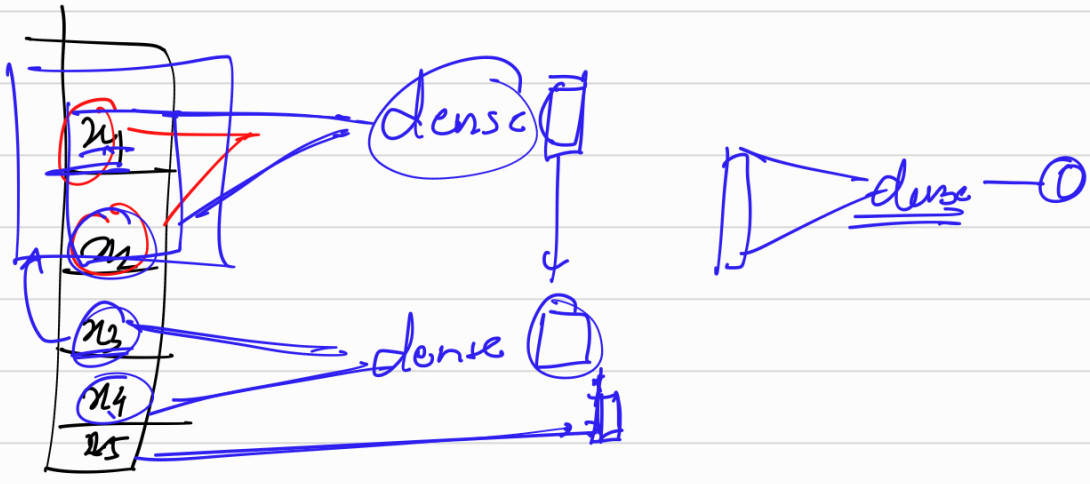
hL3 = Dense (20) (inp2).

hL4 = Dense (50) (hL3)

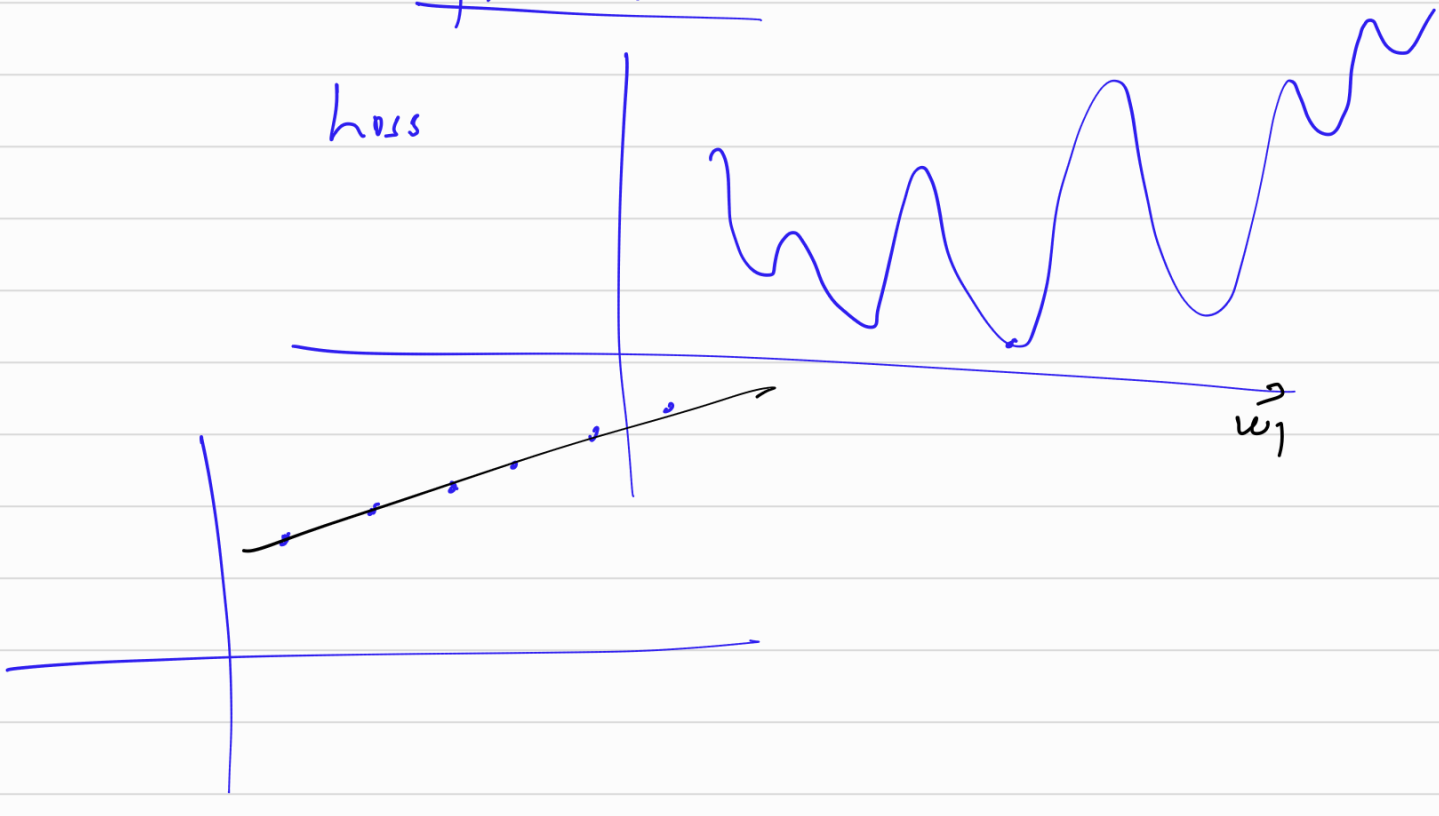
O = Concatenate ( ) ([hL2, hL4])

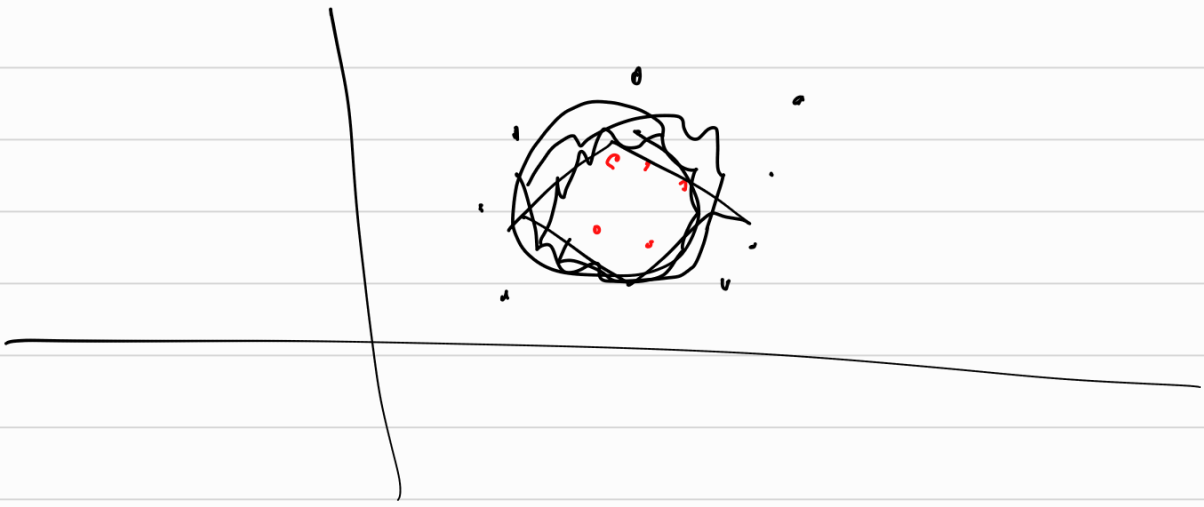


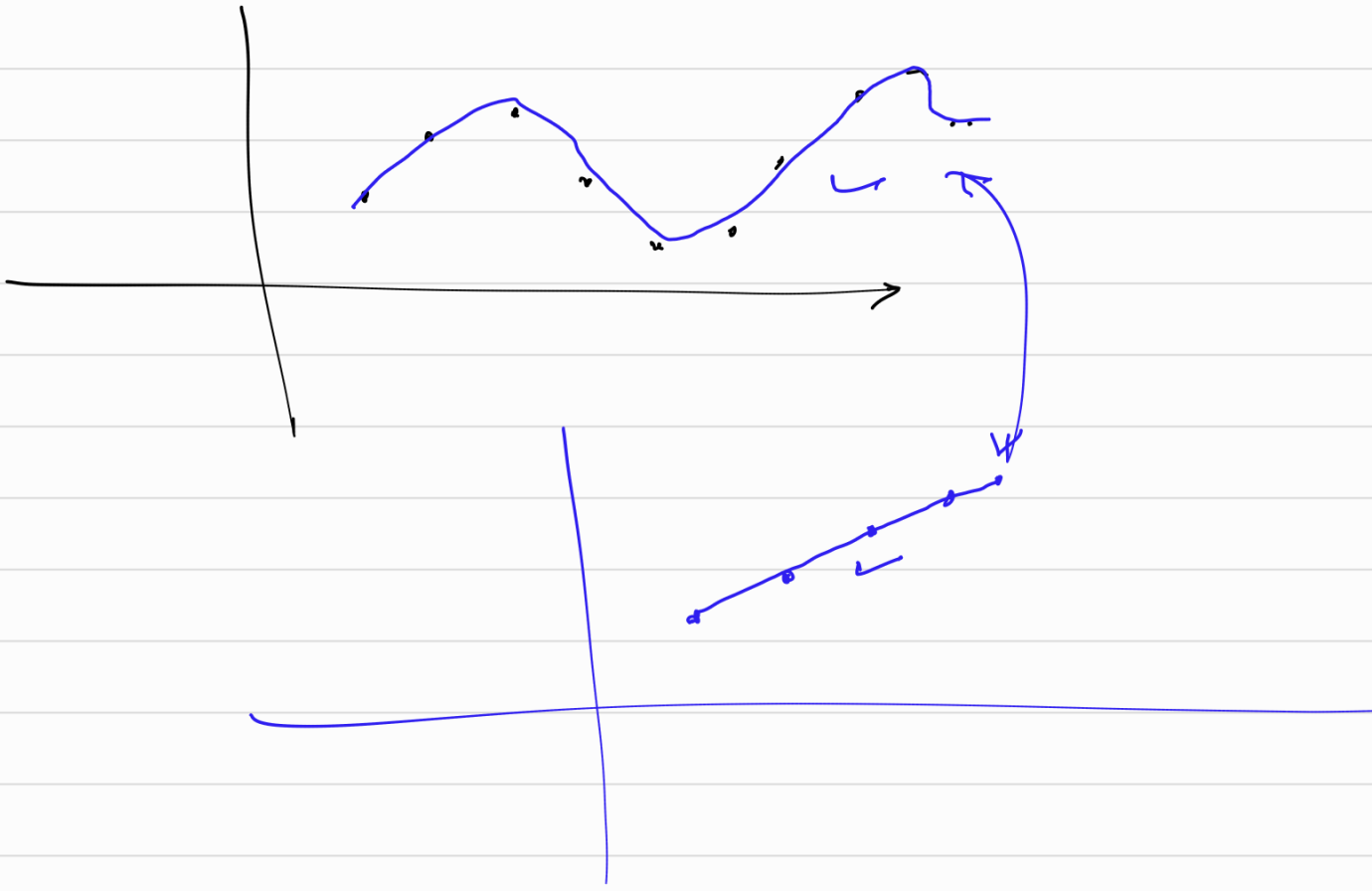

Temp
loss Reg(MSE)
MSE/cc
Rainy, Sunny, Windy. (CCE)
10
5
4
loss MC

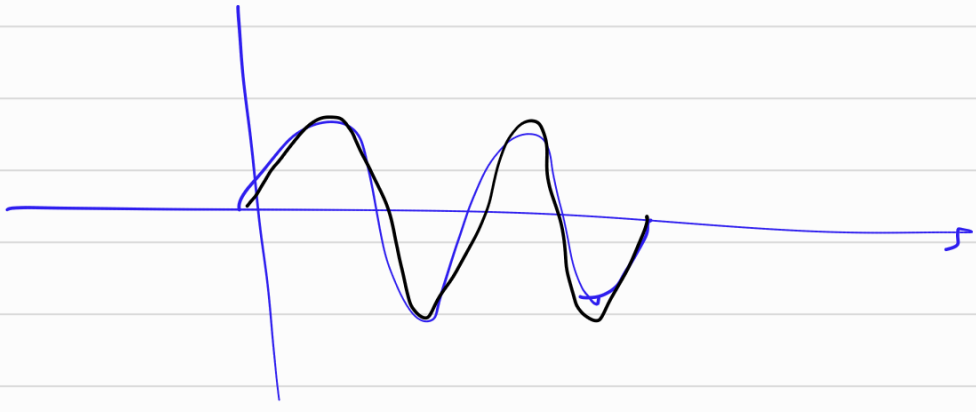

dense
dense
dense

Optimizers

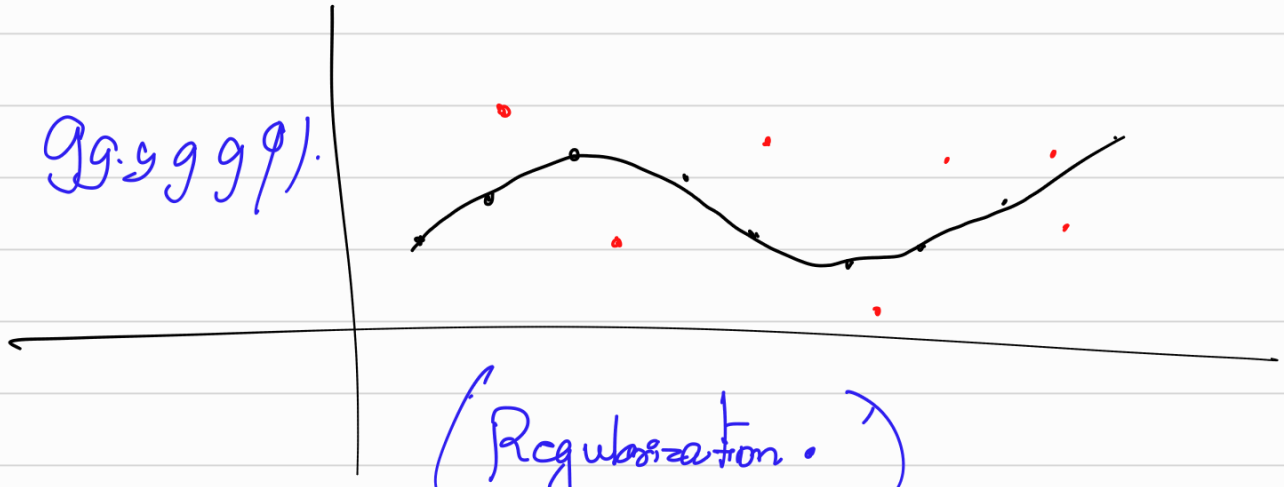

loss
w_1







99.9999%.



(Regularization.)

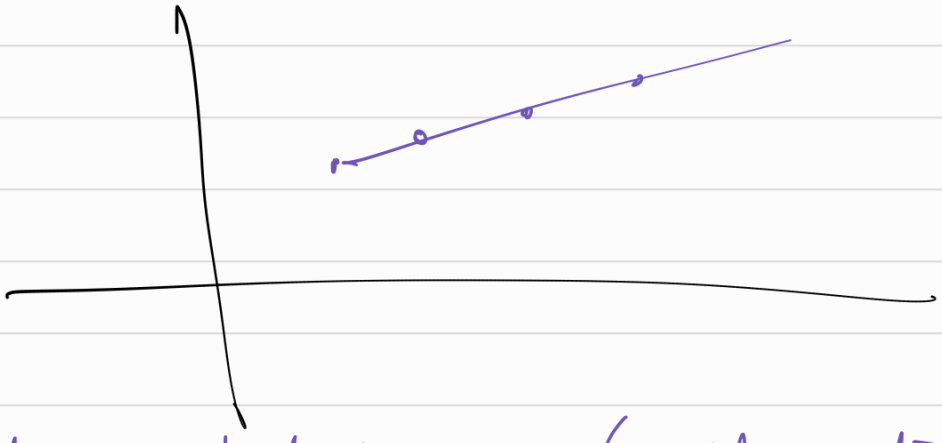
- Regularization
  - Dropout.
  - optimizers.
  - Callbacks
  - Batch Normalization
  - weight initialization

→ Hyperparameter Tuning.

Regularization:

↓↓

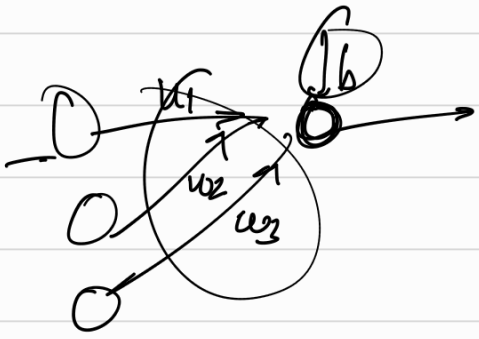
(Ridge and Lasso.) ?



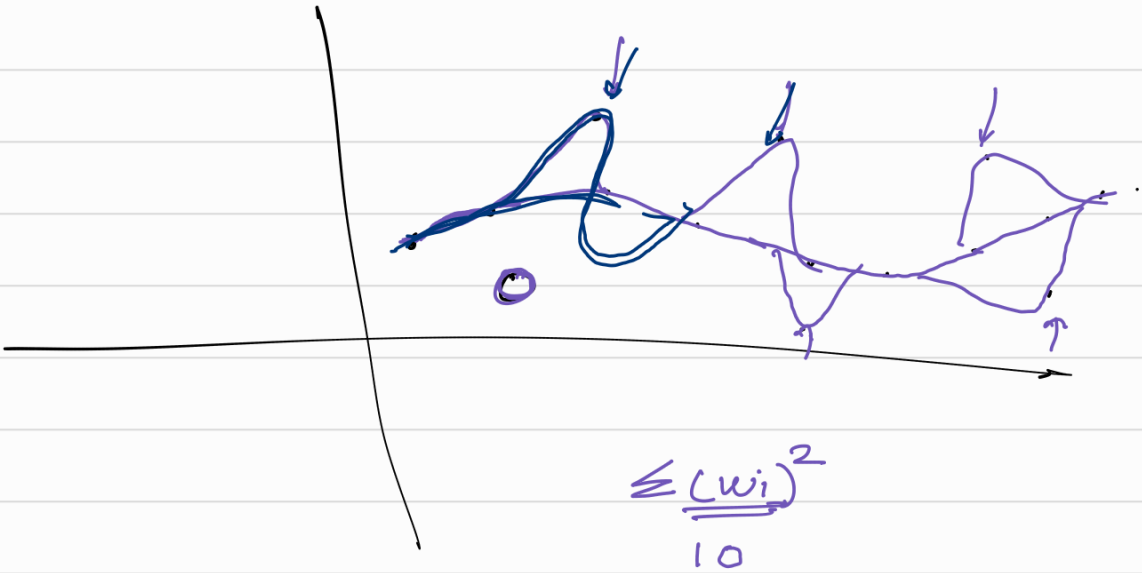
Overfit → has high variance (model is extremely sensitive to changes in w dist).

(1) $y = 1000x_1 + 500x_2$ $\quad y = 1000$

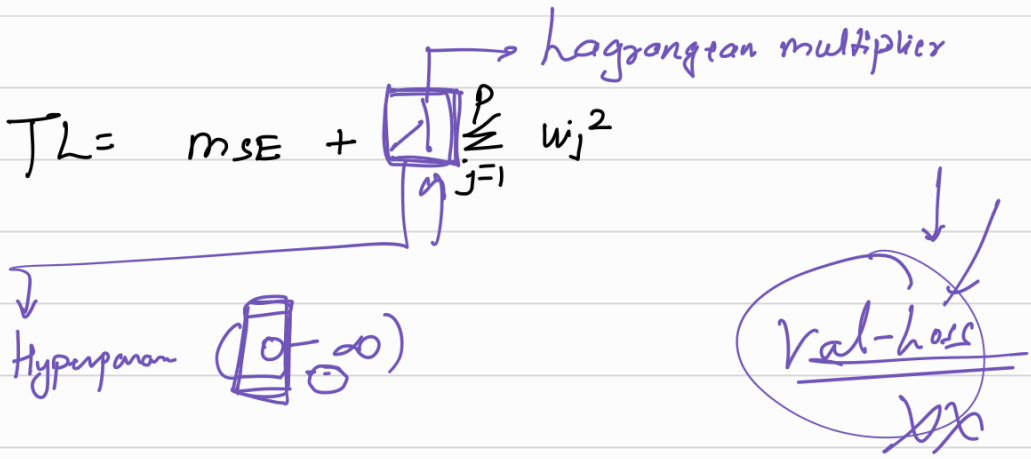
(2) $y = 2x_1 + x_2 \quad = (2)$

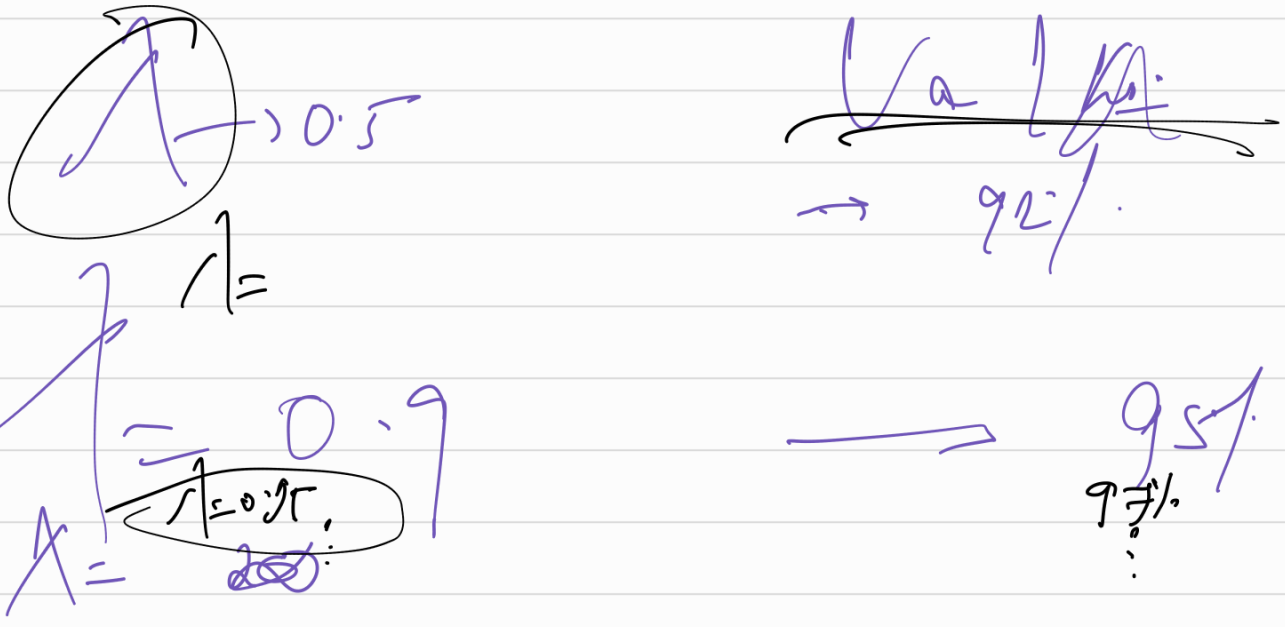


$N_1 = w_1 x_1 + w_2 x_2 + w_3 x_3 + b_1$



TL	mSE	$\sum (w_i)^2$
510 =	500	10
425 =	400	25
375 =	320	55
355 =	260	95
320 =	200	= 120
305 =	150	= 155
345 =	145	= 200


Lagrangian multiplier
$TL = mse + \lambda \sum_{j=1}^{p} w_j^2$
Hyperparam (λ, 0 ~ ∞)
Val-loss


0.5
$\lambda =$
$= 0.9$
$\lambda = 0.7$
Val loss
92%
95%
97%

Reg.

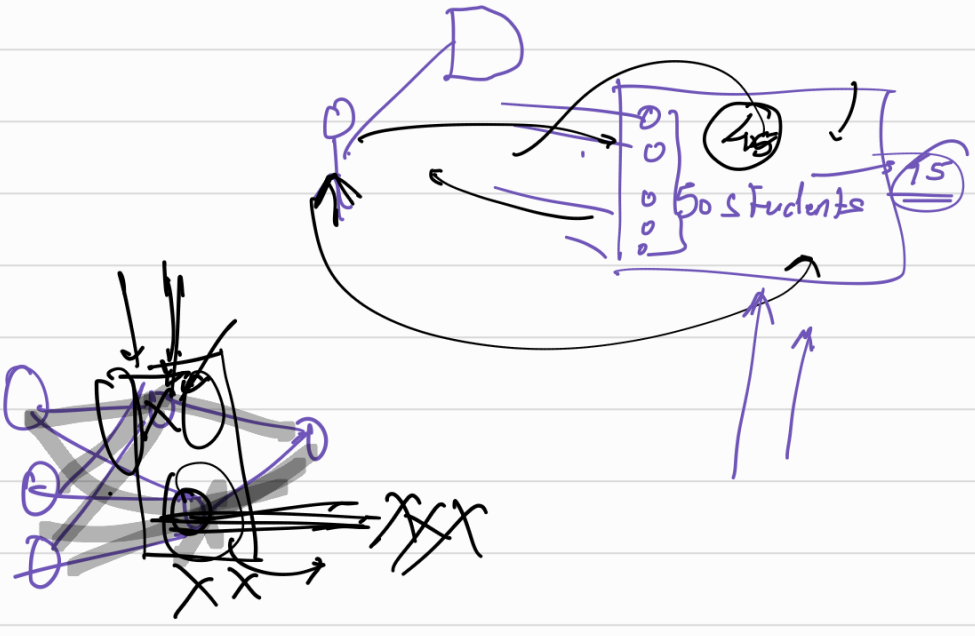
$Th = mse + \lambda \sum w_j^2$ Ridge (L^2).

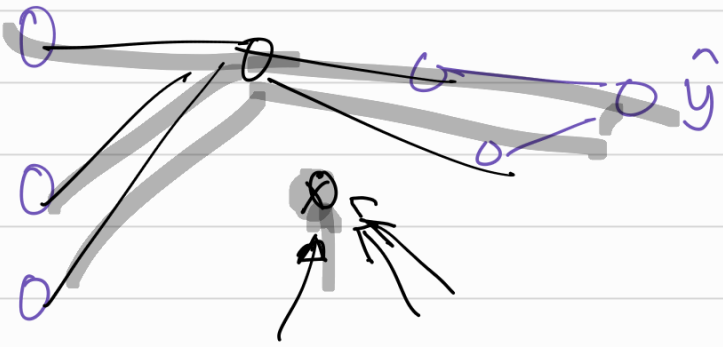
$Th = mse + \lambda \sum |w_j|$ Lasso (L_1).

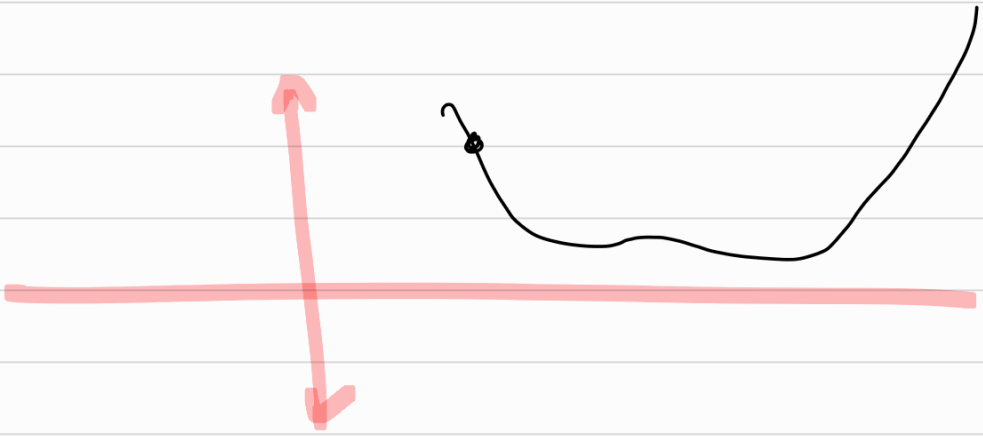
$Th = mse + \lambda_1 (\sum w_j^2) + \lambda_2 \sum |w_j|$ $L_1 L_2$ Elastic Net

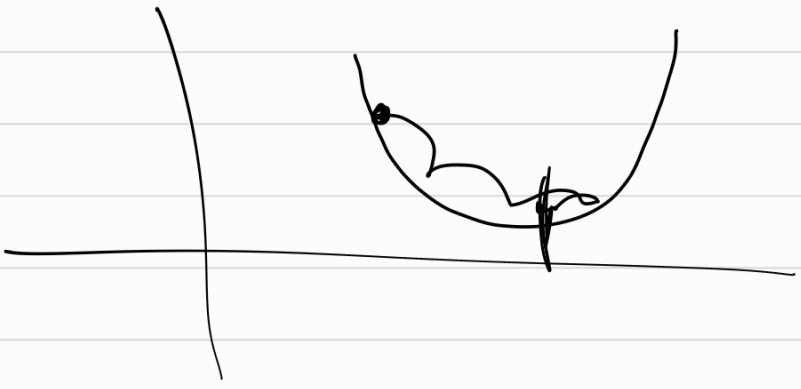
Break!

Dropout:

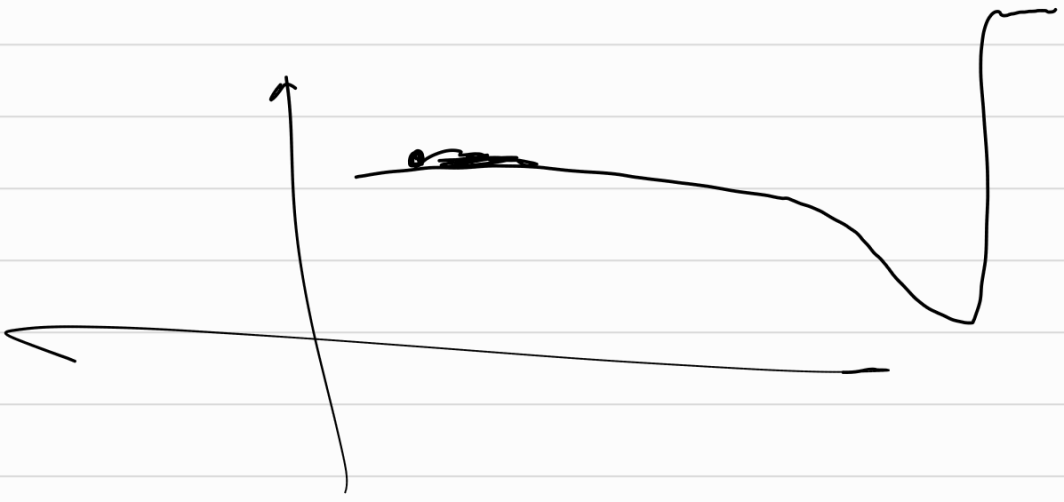




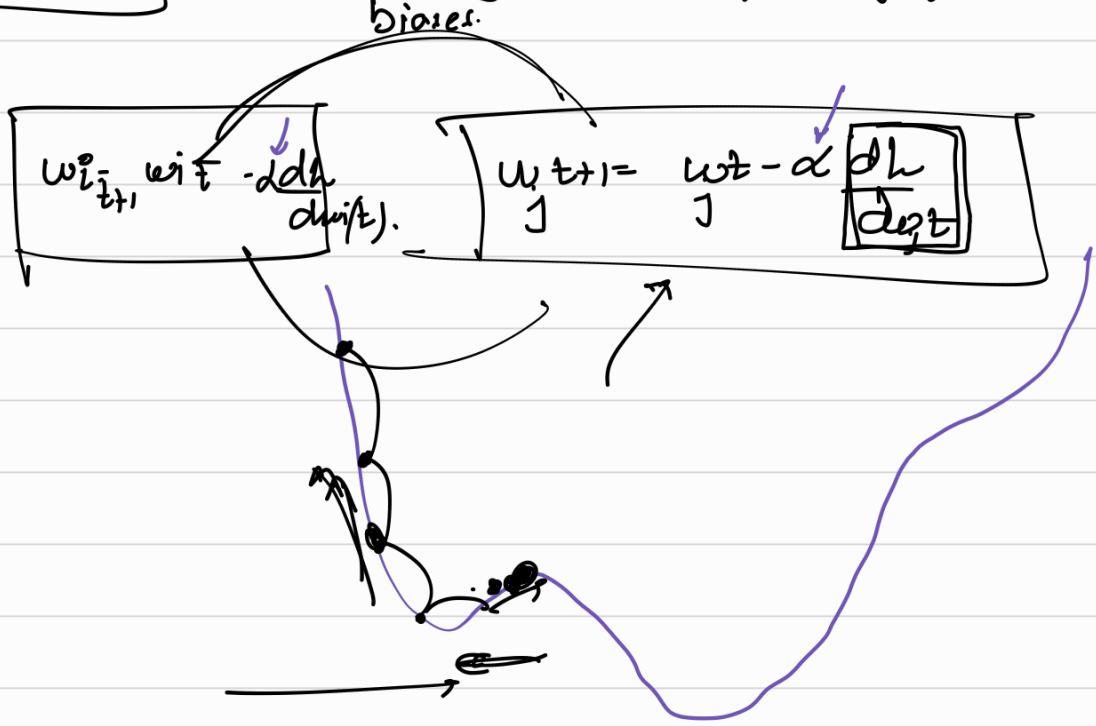


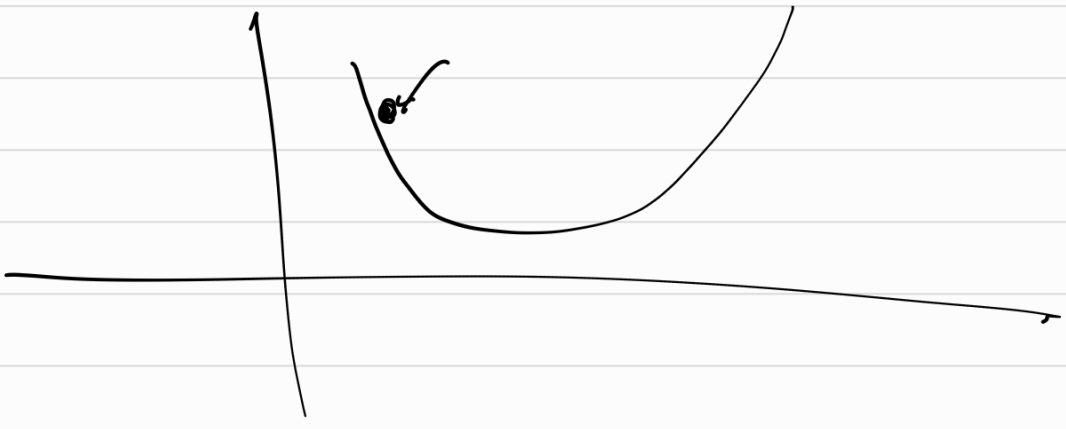


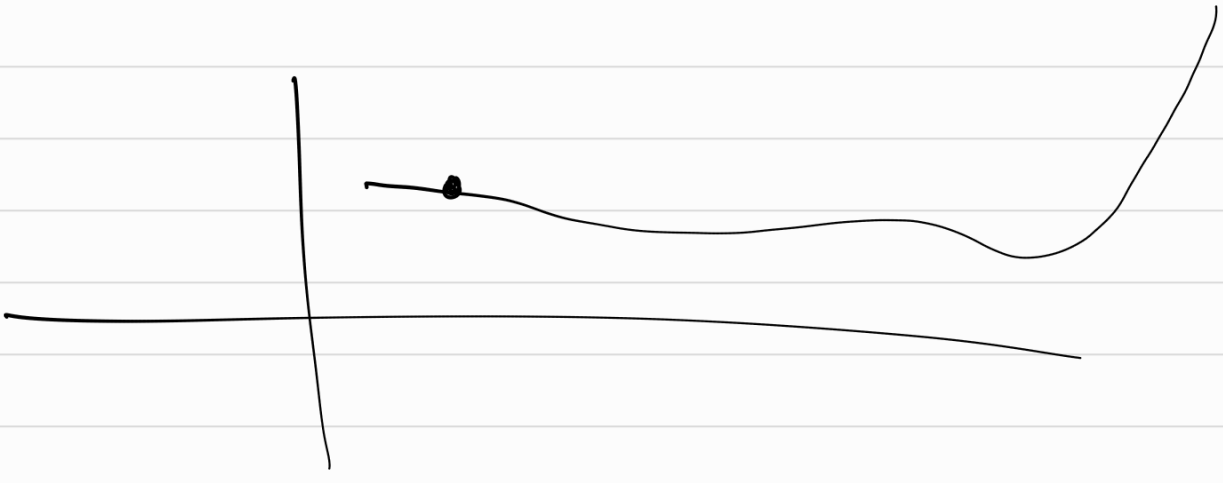
10



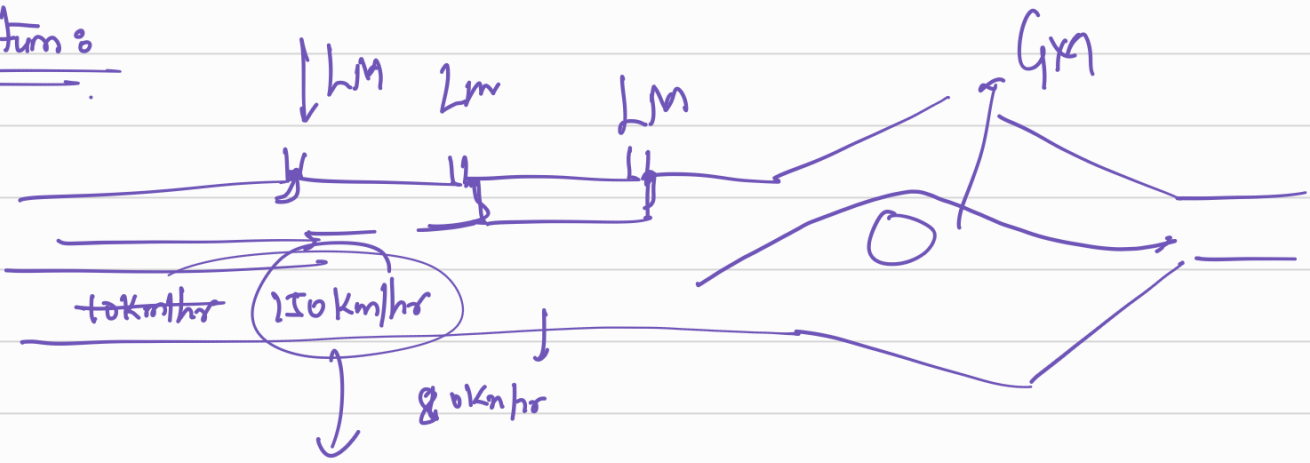
optimizers : minimizing the loss by changing the weights and biases.







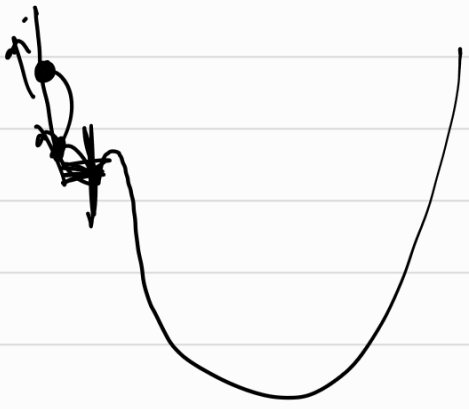
Momentum :



momentum

GD

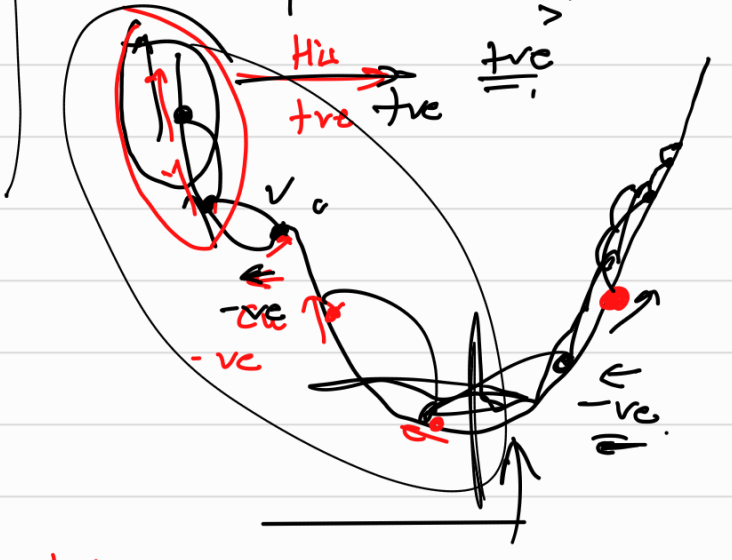
$$w_{t+1} = w_t - \alpha \frac{dh}{dw_t}$$



GD with momentum

$$w_{t+1} = w_t - update_t$$

$$update_t = \beta\, update_{t-1} + (1-\beta)\,\alpha \frac{dh}{dw_t}$$



$t \Rightarrow t=1 \;\; t=2 \;\; t=3$

$$update_1 = (1-\beta)\,\alpha \frac{dh}{dw_2}$$

$$update_2 = \beta\, update_1 + (1-\beta)\,\alpha \frac{dh}{dw_2}$$

$$update_2 = \beta\left((1-\beta)\alpha \frac{dh}{dw_1}\right) + (1-\beta)\frac{dh}{dw_2}$$

$\alpha \beta < 1$

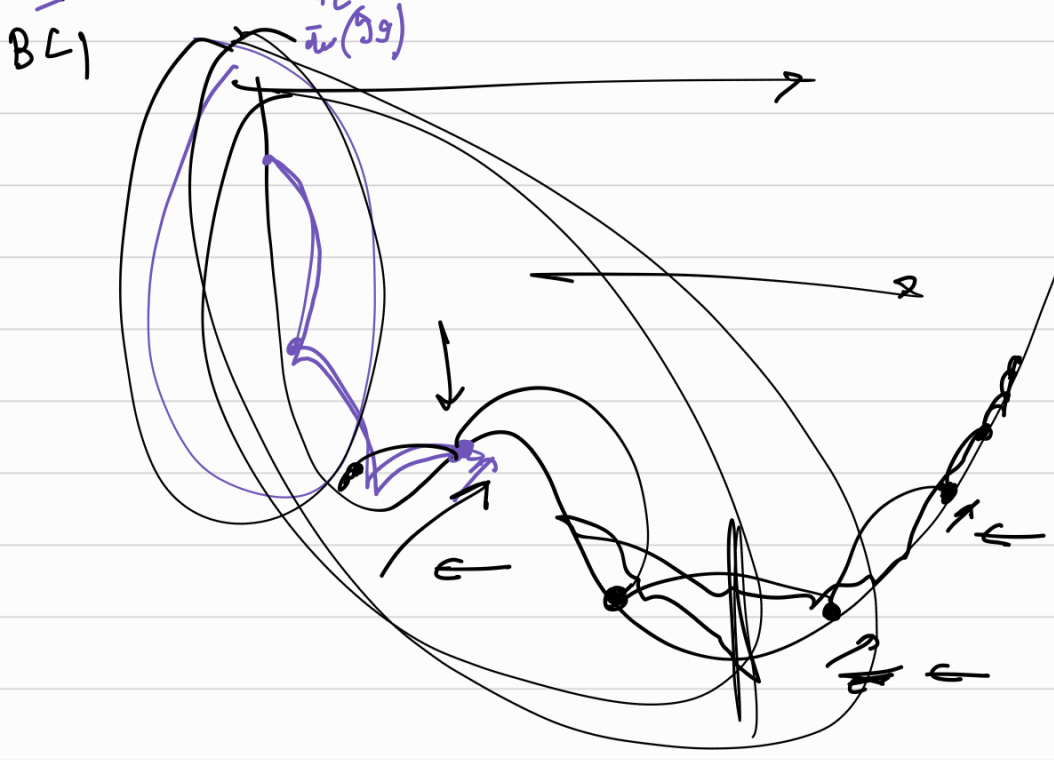
$$update_3 = \beta\, update_2 + (1-\beta)\,\alpha \frac{\partial h}{\partial w_3}$$

$$update_3 = \beta\left(\beta(1-\beta)\alpha\frac{\partial h}{\partial w_1} + (1-\beta)\alpha\frac{\partial h}{\partial w_2}\right) + (1-\beta)\alpha\frac{\partial h}{\partial w_3}$$

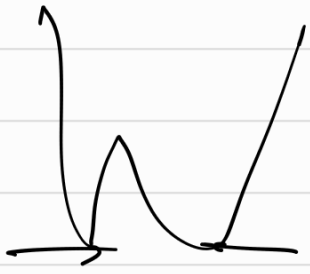
$\beta \cdot \alpha$

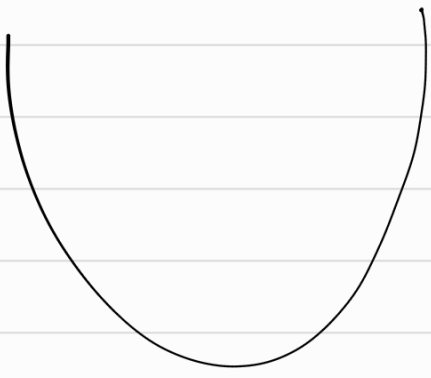
$$update_3 = \beta^2(1-\beta)\alpha\frac{\partial h}{\partial w_1} + \beta(1-\beta)\alpha\frac{\partial h}{\partial w_2} + (1-\beta)\alpha\frac{\partial h}{\partial w_3}$$

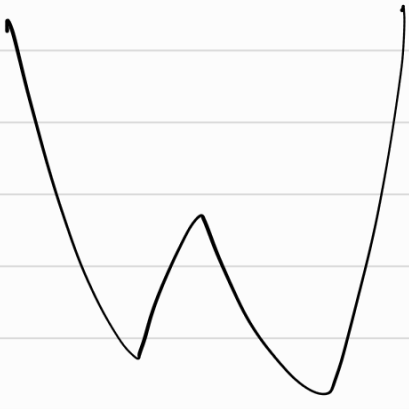
$\beta < 1$

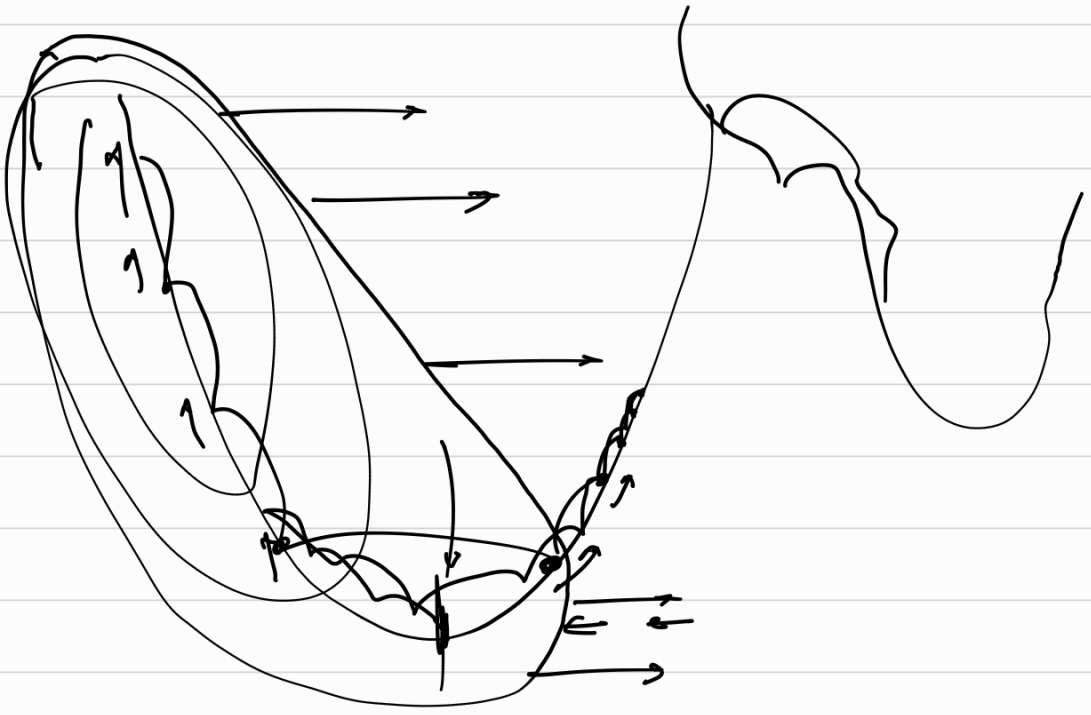


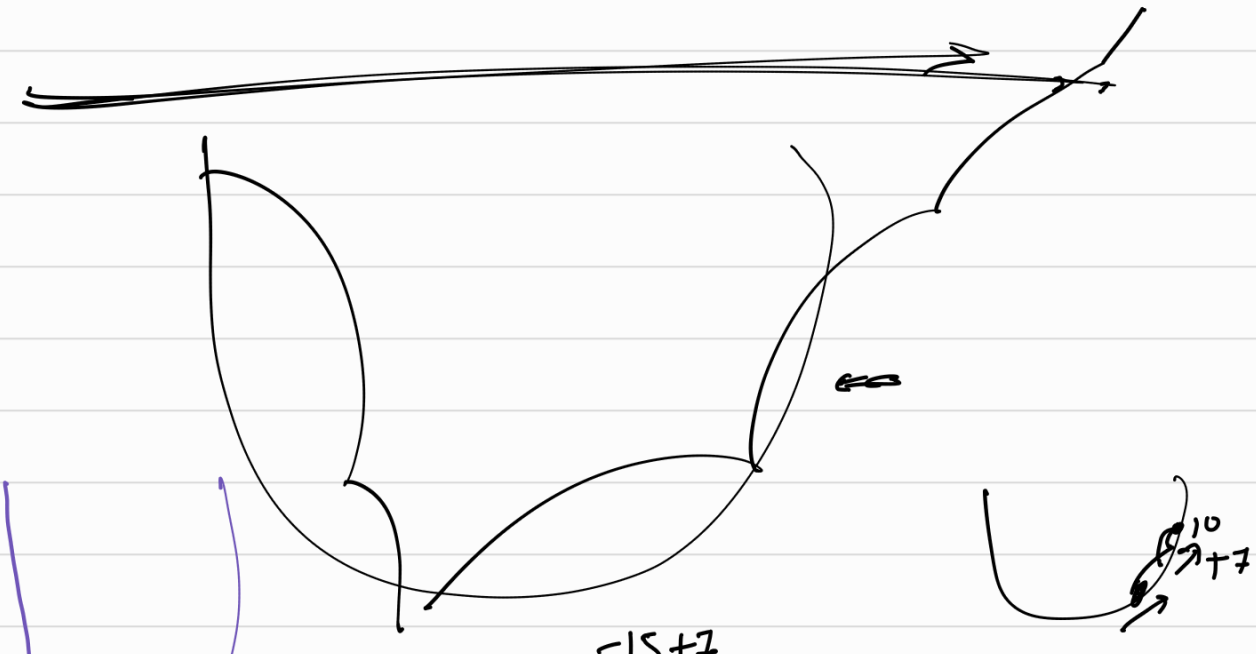
$i = i + 5$











$-15+7$

$-20 \rightarrow -20+5$

$-ve$

$$\text{update}_t = \text{update}_{t-1} + \alpha \frac{dh}{dwt}$$

1

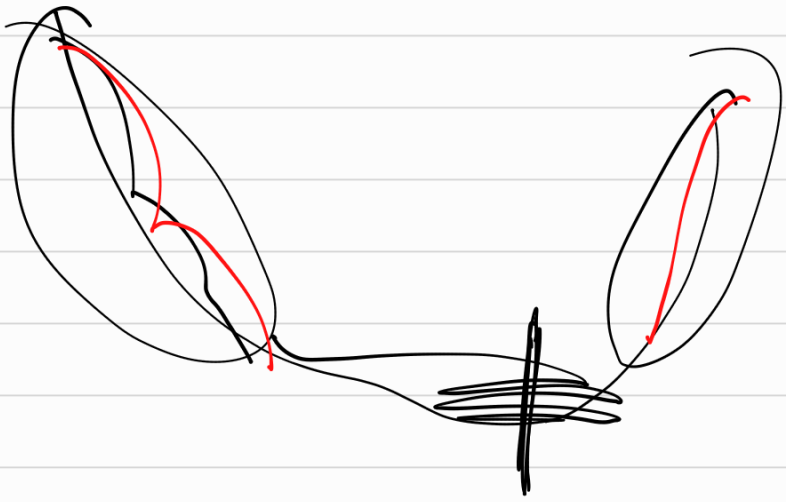
2

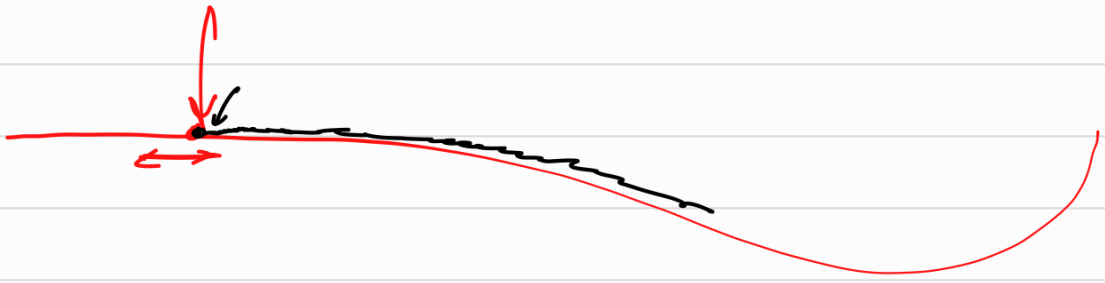
3

$-8 \quad +10$

$= -15$

6th





Momentum → GD.

$a_0 = 0$

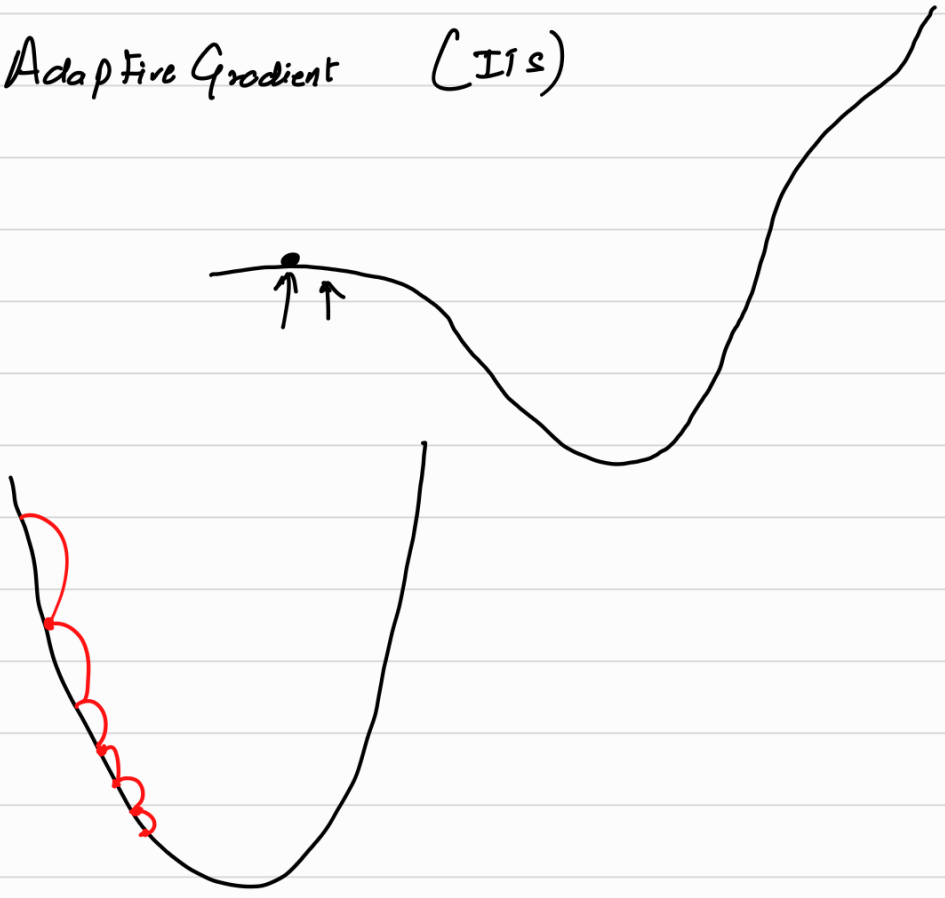
$a_t = a_{t-1} + 5$

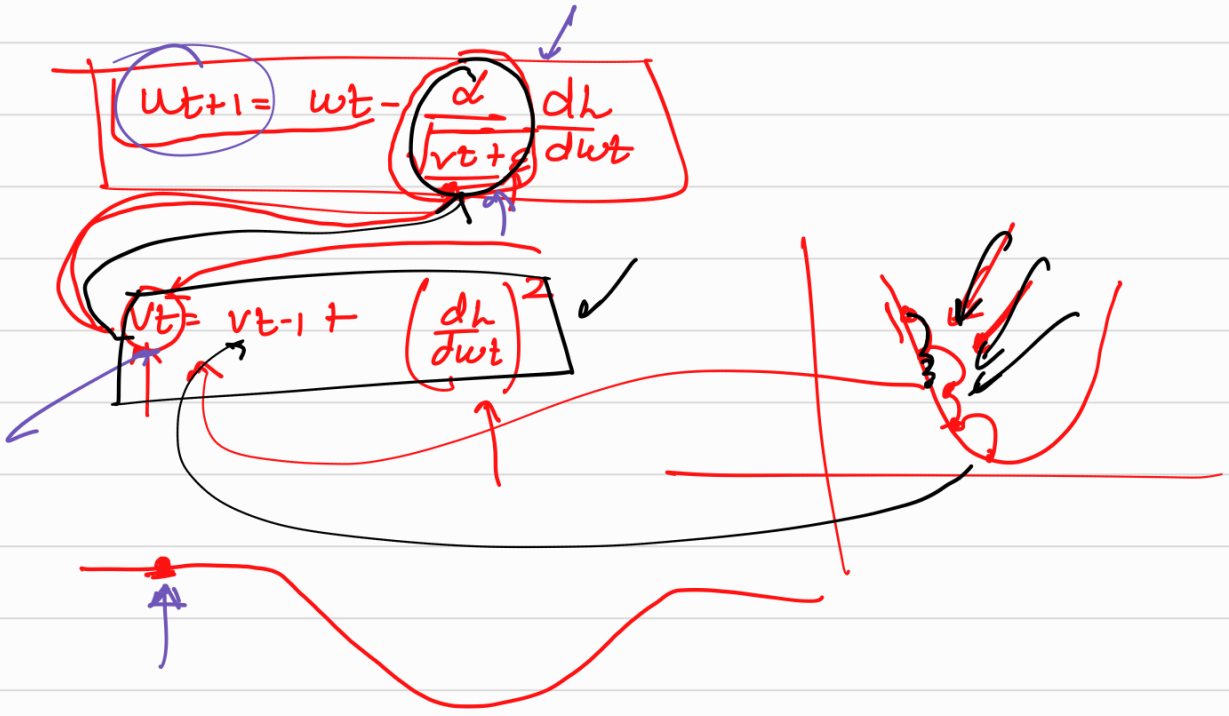
$a_1 = 0 + 5$ $\quad a_2 = a_1 + 5$ $\quad a_3 = a_2 + 5$

$a_1 = 5$ $\quad a_2 = 10$ $\quad a_3 = (a_1 + 5) + 5$

Break!

Adagrad: Adaptive Gradient (IIs)





Rms Prop

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

$$v_t = \beta \, v_{t-1} + (1-\beta) \times \left(\frac{\partial h}{\partial w_t}\right)^2$$

0.99 , 0.01

Adam:

Adaptive moments →

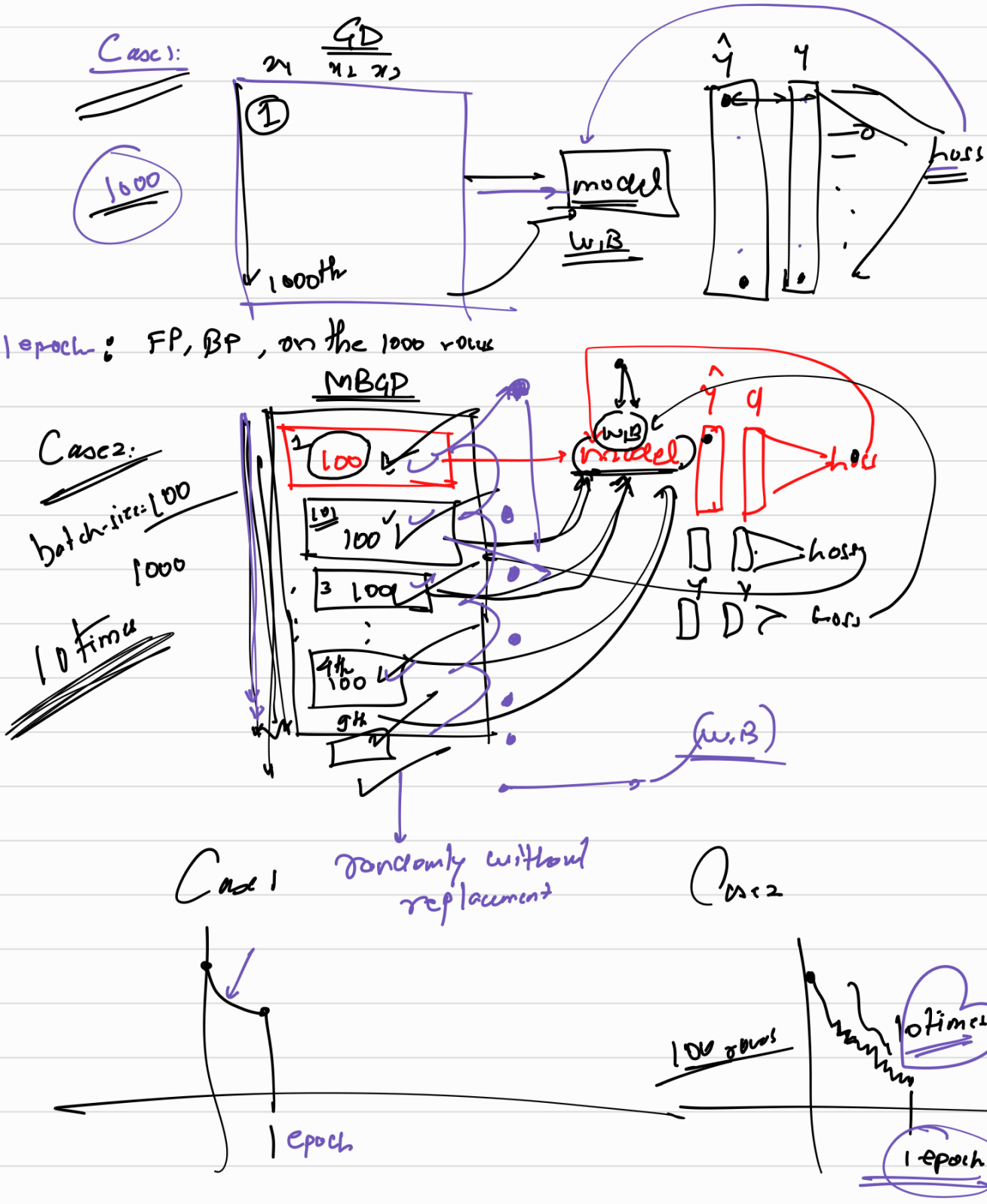
$$m_t = \beta_1 (m_{t-1}) + (1-\beta_1) \frac{\partial h}{\partial w_t}$$

$$v_t = \beta_2 (v_{t-1}) + (1-\beta_2) \times \left(\frac{\partial h}{\partial w_t}\right)^2$$

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{v_t + \varepsilon}} \times m_t$$

Mini Batch Gradient Decent → Batch Normalization

Callbacks



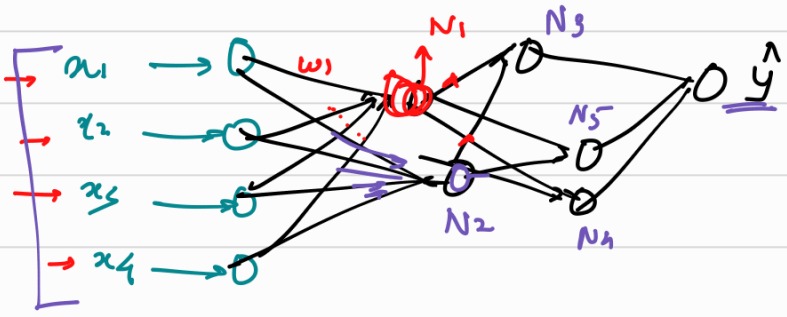
model.fit (10)

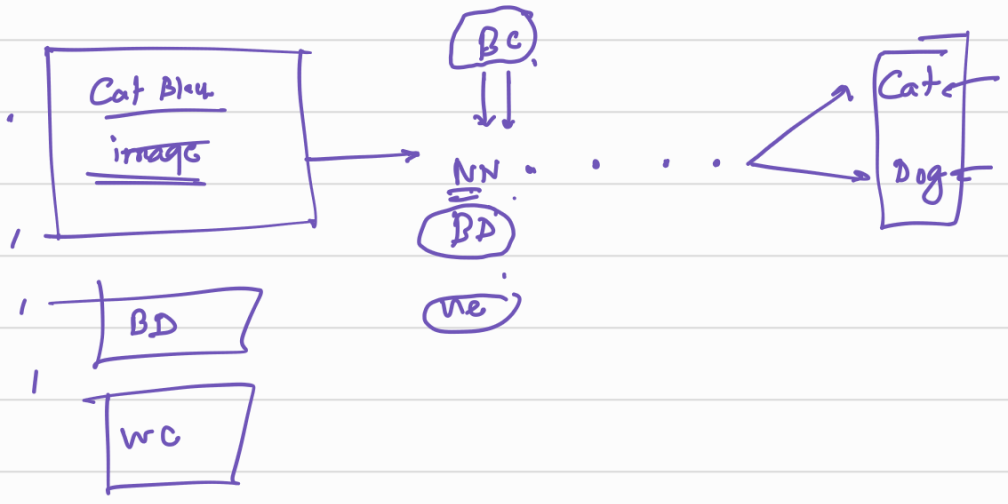
model.fit (20)

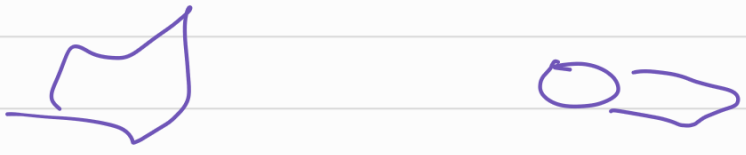
Batch Normalization

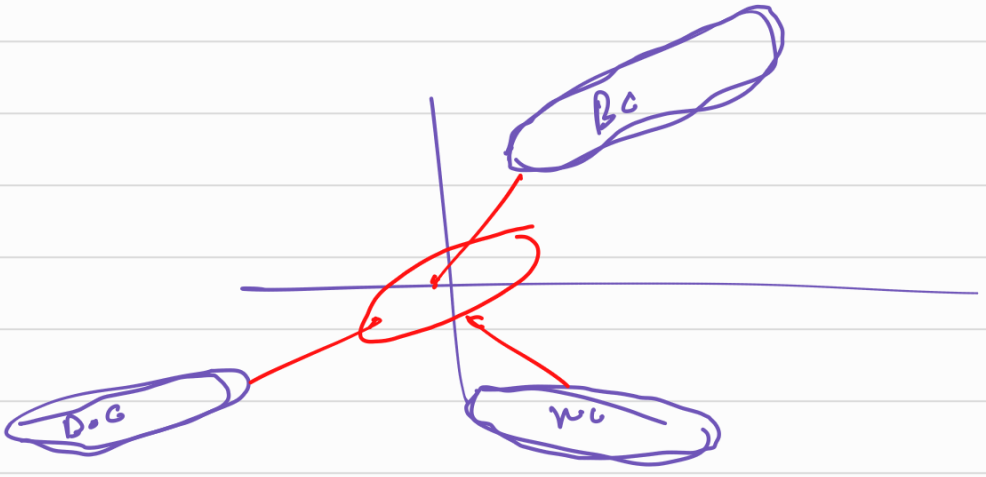
Callbacks

Batch Normalization

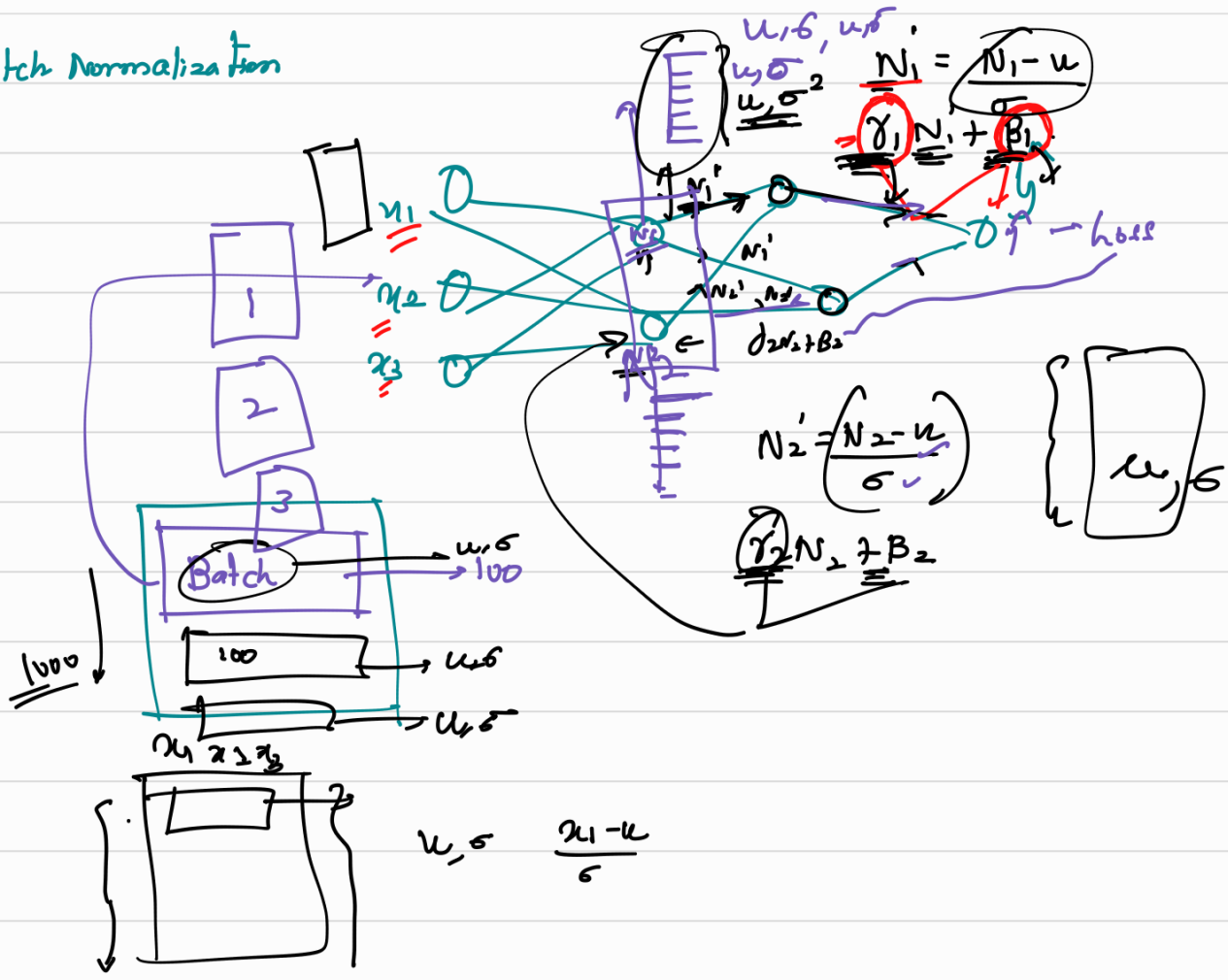


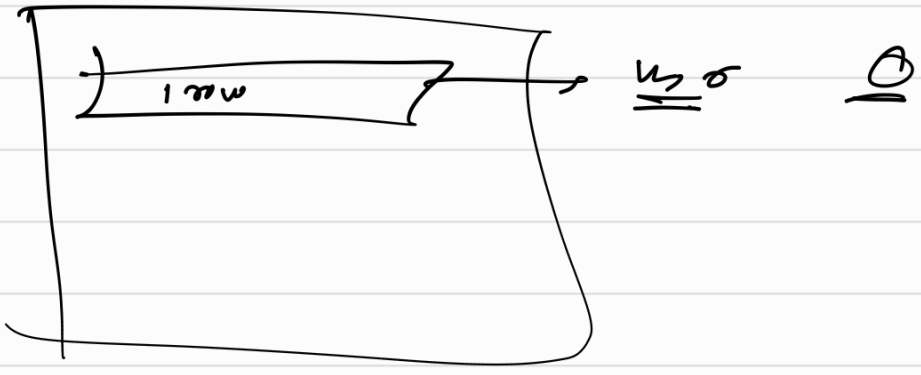


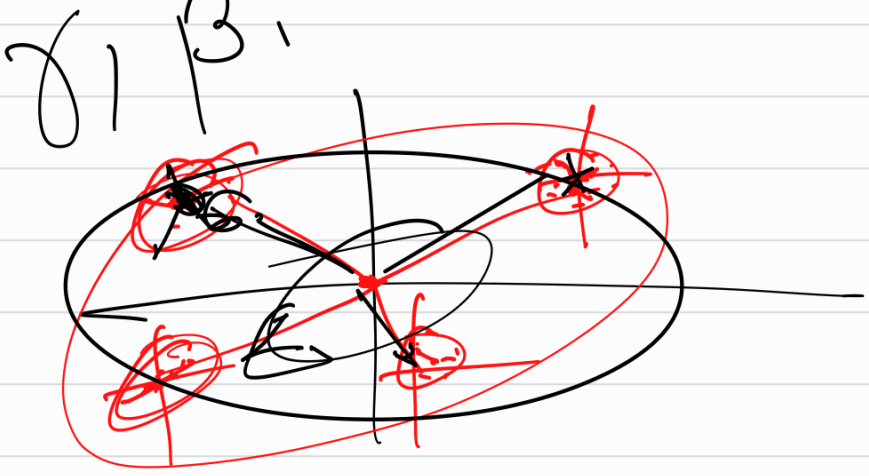


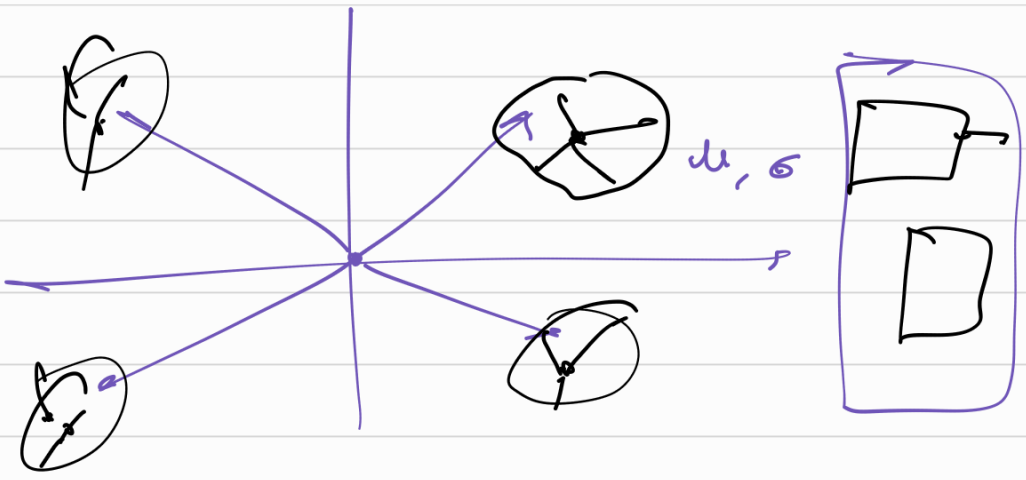


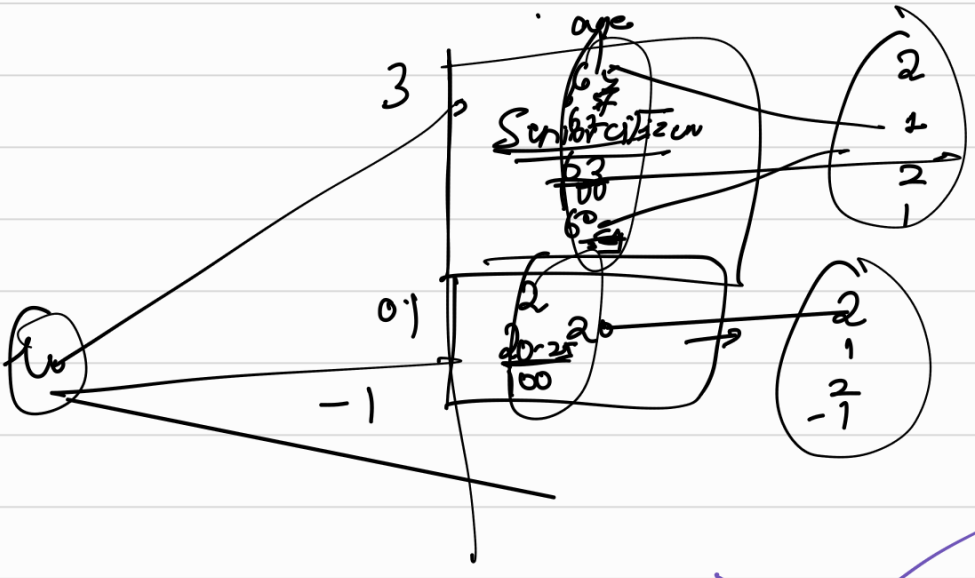
Batch Normalization

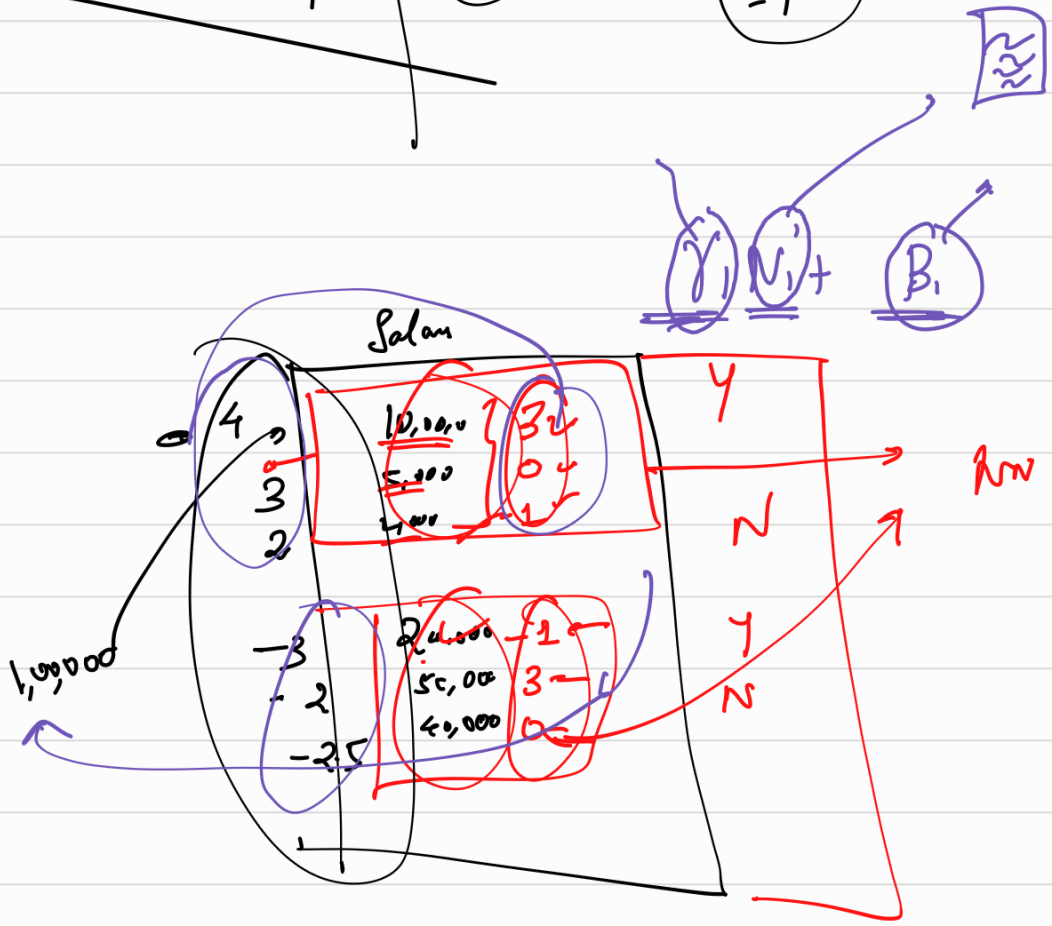


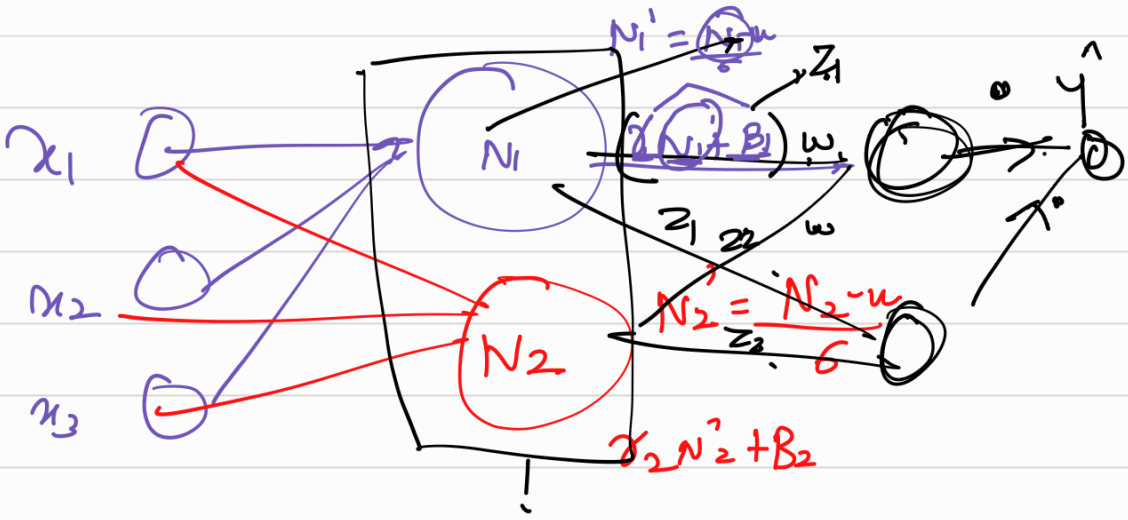




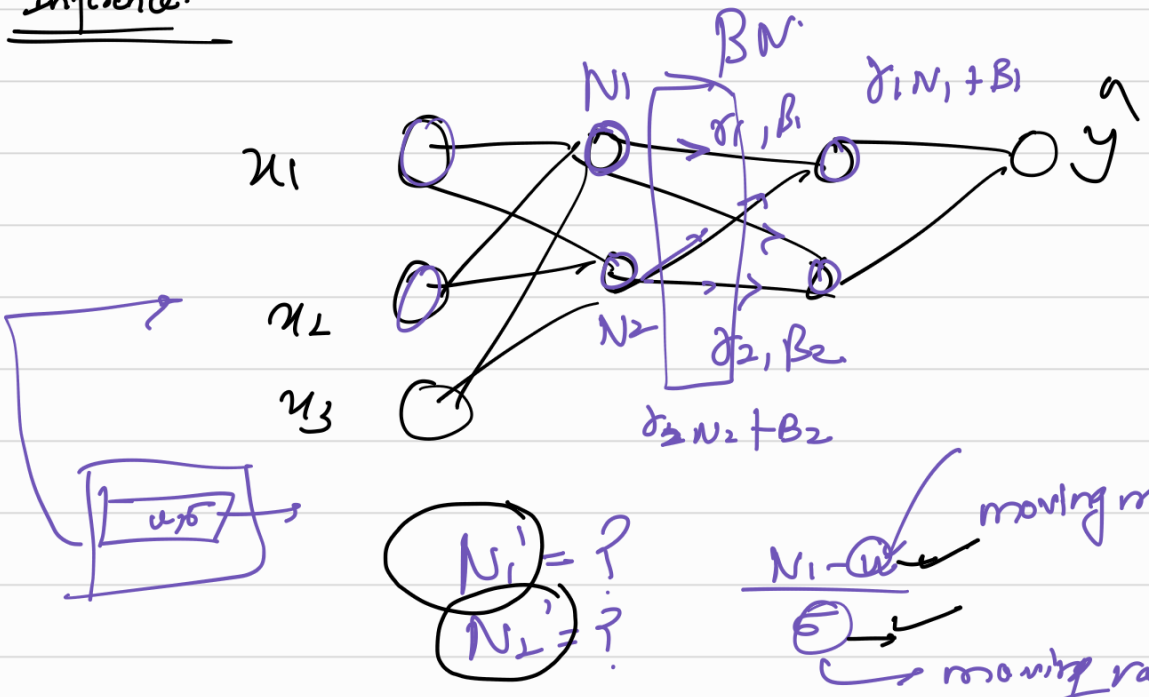

μ, σ

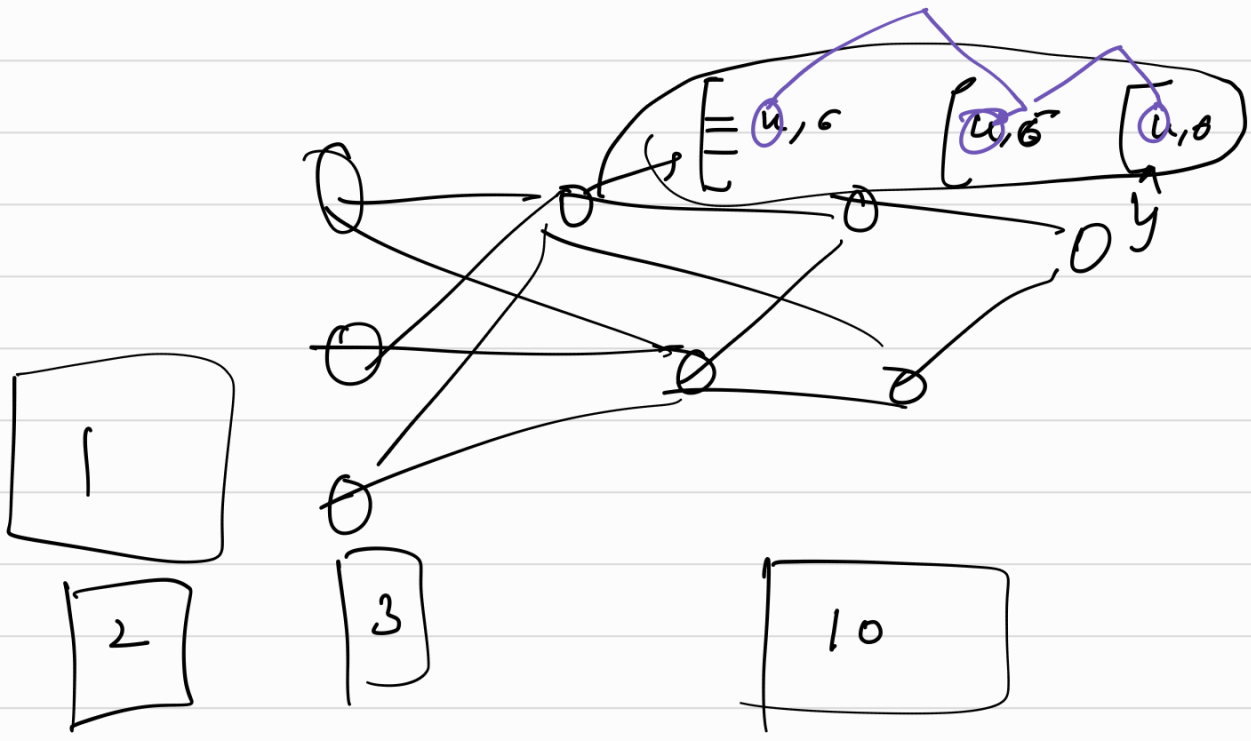

age
3
Senior citizen
0.1
-1
2
1
2
1
2
-1


Salary
4
3
2
-3
-2
-2.5
1,00,000
10,000
5,000
4,000
40,000
50,000
Y
N
Y
N
Any


x_1
x_2
x_3
N_1
N_2
$N_1' = \frac{N_1 - \mu}{\sigma}$
$\gamma_1 N_1' + B_1$
$N_2' = \frac{N_2 - \mu}{\sigma}$
$\gamma_2 N_2' + B_2$
z_1
z_2
z_3
ω_1
ω_2
ω_3
y

Inference:


BN
N_1
N_2
γ_1, B_1
γ_2, B_2
$\gamma_1 N_1 + B_1$
$\gamma_2 N_2 + B_2$
x_1
x_2
x_3
y
μ, σ
$N_1' = ?$
$N_2' = ?$
$\frac{N_1 - \mu}{\sigma}$
moving mean (-bair)
moving var (bair)

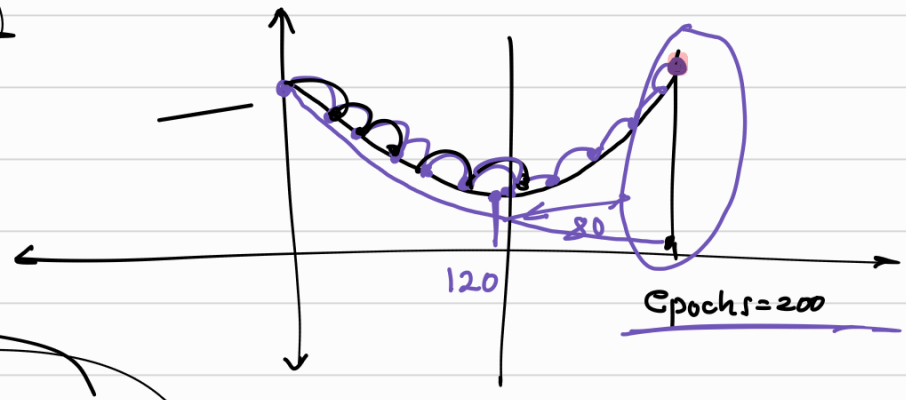

μ, σ
μ, σ
μ, σ
y
1
2
3
10

N1

moving mean → β × moving mean + (1-β) × mean(batch)

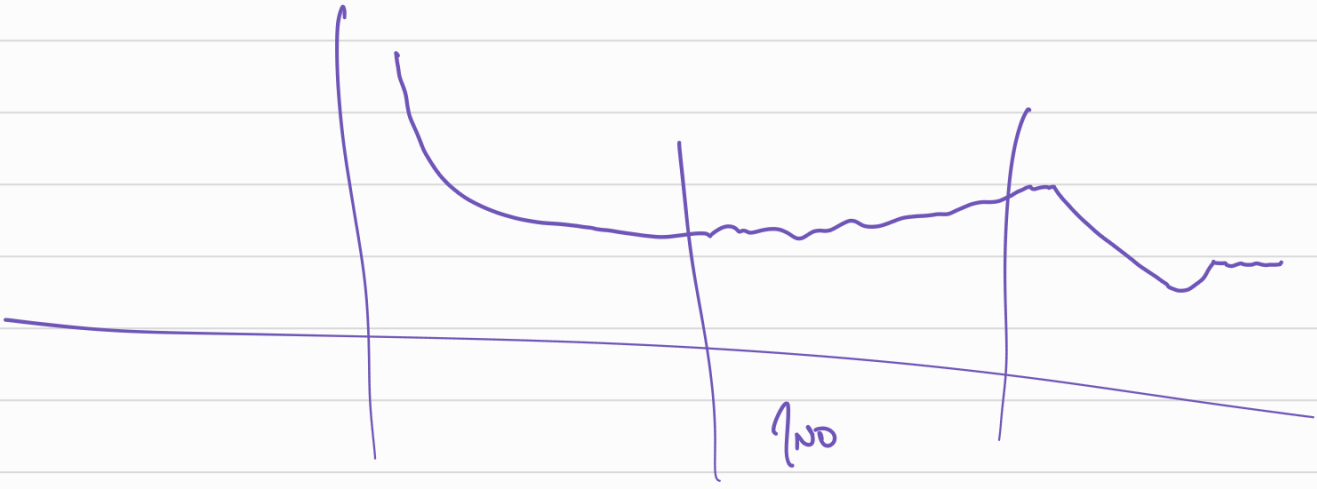
N2 moving var = β × moving var + (1-β) var(batch)

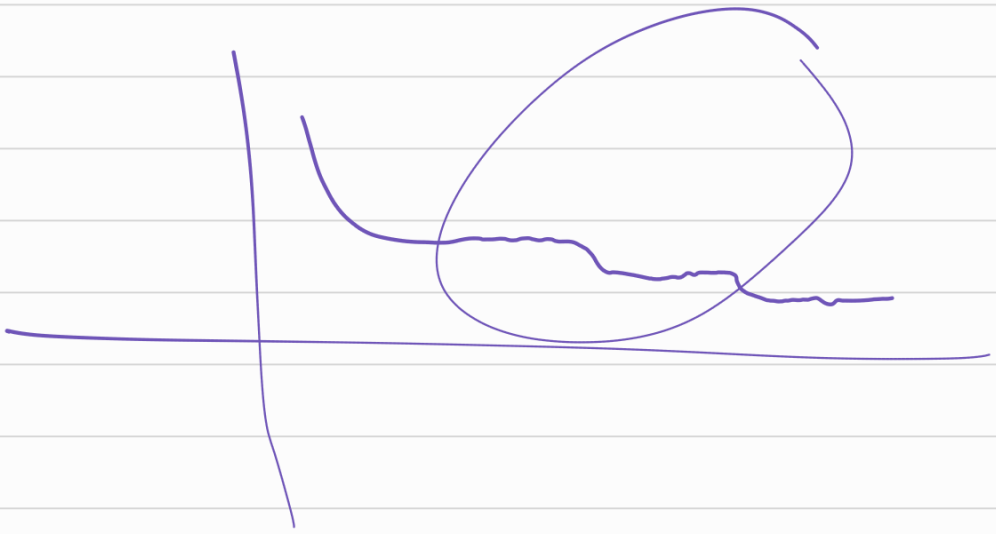
Callbacks



model checkpoint

Early stopping





max_trials = 3

Executions_per_trial = 2

[3, 32, relu, d=T, lr=10^{-2}] → model → fit model. (100)

model → fit (100)

[4, 32, relu, d=N, lr=10^{-4}]

[4, 64, tanh, d=T, lr=10^{-3}]

11:33 pm

Input layer (784)

{Hidden layer (200, 400, activation (tanh, relu).

Hidden layer (100, 200 activa tanh, relu).

2,4 {[Batch Norm() Hidden (50, 100, activation, (tanh, relu), λ = 0.1, 0.5]} ← log sampling

Batch Norm() ← maybe add / not

model.layers (Dense (10, softmax).