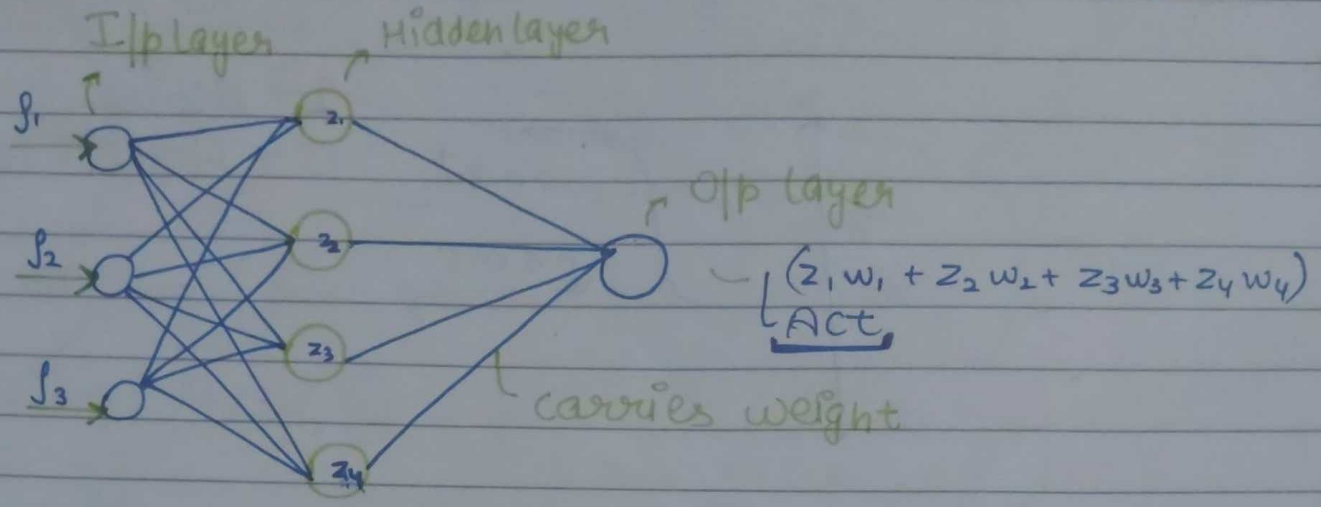
NN

29/09/22

Nodes

- eyes, skin, ear → I/p layer | 'Neurons' → Hidden layer

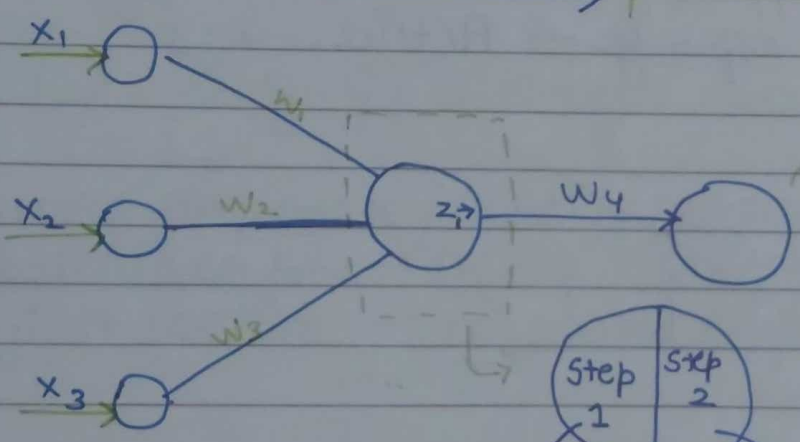
=



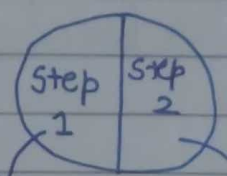
↓ Simplified version to understand Better

'Forward Propagation'

=



O/p layer output
↓
$Z_0 = Z_1 \times W_4 + \ldots$
$Z_0 = \text{Act. func}(z)$



similar steps

Step 1 → $y = w_1 x_1 + w_2 x_2 + w_3 x_3 + \text{Bias}$

Step 2 → $Z_1 = \text{Act. func}(y)$

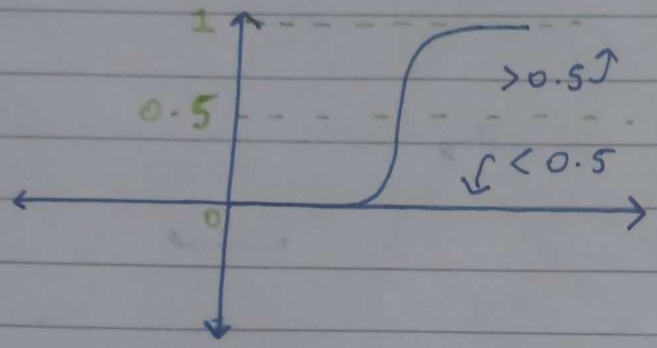
↳ ex: sigmoid → $\frac{1}{1+e^{-y}}$ – The value → active or not

Act func → To activate which Neuron will forward the Response (like → hot piece on left hand will activate Neurons of left hand and not Right hand)

★

• 1) Sigmoid AF → B/w 0 to 1 (To activate and act as output)

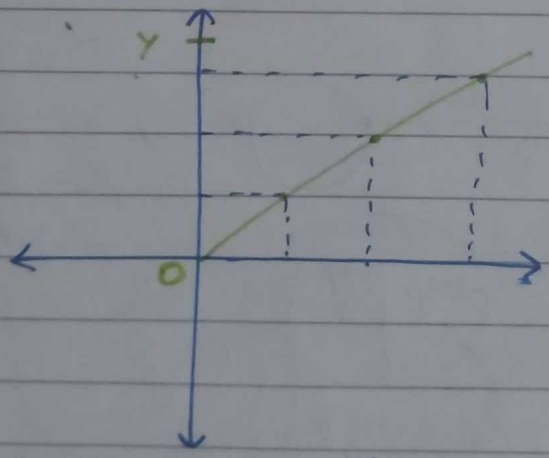
→ $\frac{1}{1+e^{-y}}$ [Set as Threshold >, <]



1 → Active
0 → Active - Non.

2) ReLu AF → max(y, 0) [No -ve values]

- if y = -ve → o/p = 0 → Non - Active
- if y = +ve → o/p = y → Active.

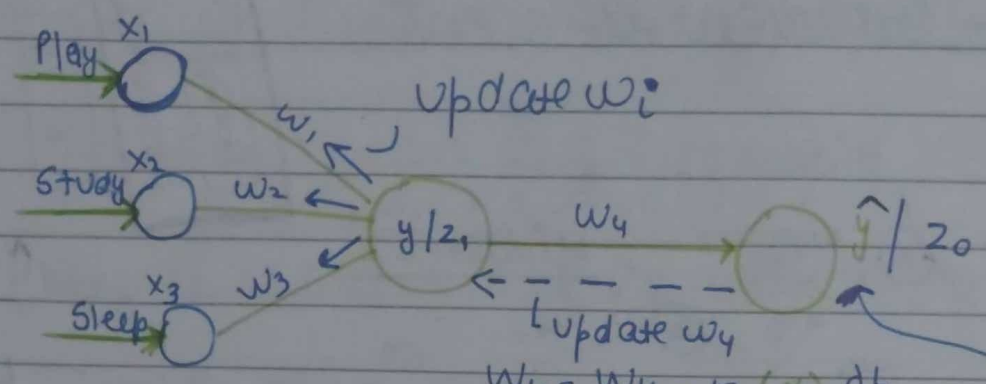


X — X — X — X — X —

☆ NN TRAINING → Back Prop + Loss func

Play	Study	Sleep	O/p
2h	4h	8h	1 - pass → y

- Forward Prop → Act func
- Backward Prop → optimizer + loss func^n



$$W_4 = W_4 - \eta(\alpha)\frac{\partial L}{\partial W_4}$$

$$y, z_1 = [x_1 w_1 + x_2 w_2 + x_3 w_3] + \text{Bias}$$

$$z_1 = Act(y) \qquad \left| \begin{array}{l} z_0 = z_1 w_4 + \dots \\ z_0 = Act(z_0) = \hat{y} \end{array} \right|$$

★ To check the accuracy of Result → compare y and $\hat{y}$

By loss func → Optimize → Minimize loss

Loss func → $\sum(y-\hat{y})^2$ (Normal loss func^n) (1)

- if wrong propagate / Classify / error huge → Define Optimizer func → Define loss func → Back track → Adjust weight → Forward Propagate. cycle repeat until epochs.

- updated $w_4 = \boxed{w_4 - \alpha \cdot \frac{\partial L}{\partial w_4}}$

  small 'α' → To reach global minima

- updated $w_3 = \boxed{w_3 - \alpha \cdot \frac{\partial L}{\partial w_3}}$

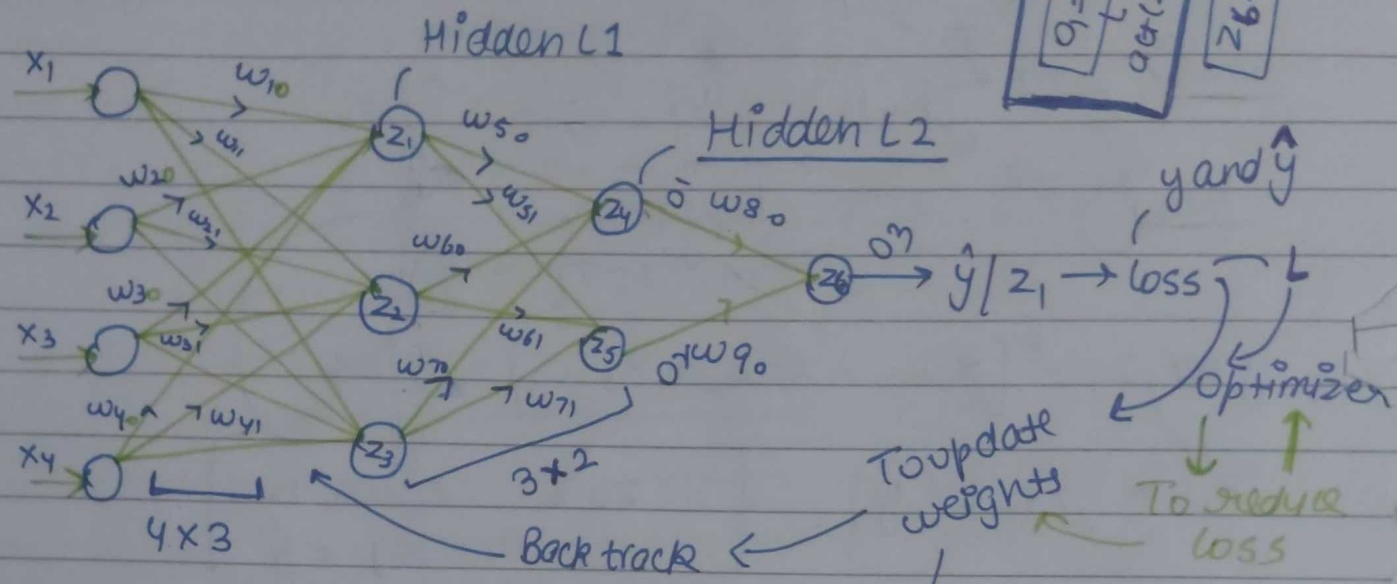
$w_{50} = w_{51}$ similar but diff

Output → Z_1 → Act(y) · weight assign

☆ Optimizer → Gradient descent

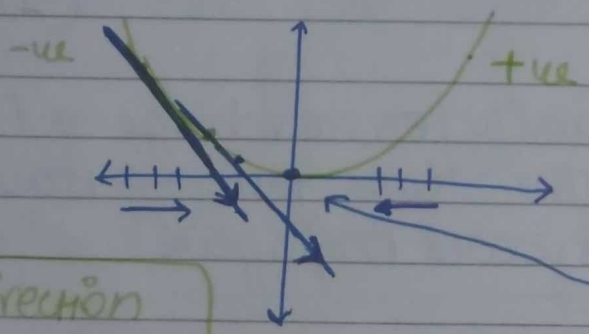
$a_1 = z_4$ | $a(z_1 w_1 + z_2 w_2)$ | $z_6 = a_3$

→



→ Updated $w_i^0 = w_i^0 - \alpha \cdot \underbrace{\frac{\partial L}{\partial w_i^0}}_{\text{slope}}$ —— ①

GD → update weights → To reach global optimal sol^n



[↘ -ve slope
↗ +ve slope]

[↘ ↙] what we want

Slope → Direction
L.R → Movement → L.R determine the movement and new updated weight.

---

if Slope = -ve → updated $w_i^0 > w_i^0$ (from ①) → Move Down (→)

if Slope = +ve → updated $w_i^0 < w_i^0$ (from ①) → Move Down (←)

- After some epoch → if loss ↓ → ✓
- After some epoch → if loss ↑ or Doesnot ↓ → x → prob with α.

•→ Chain Rule in Back Propagation

→ $z_1 . w_{50}$ → output O_1 → $z_1 . w_{51}$ + $z_2 . w_{60}$ + $z_3 . w_{60}$ Ka whole act from ~~with~~ weight (act) Σ of z_1, z_2 and z_3 and z_4 and z_5 (w_{50} similar same to w_{51})

→ $z_4 . w_{80}$ → Output O_3 (O_3

↓ Loss func → L

updated weight w_{80} → old w_{80} − $\alpha \frac{\partial L}{\partial w_{80}}$ [w_{80} updates or affects the Result z_6]

$$\frac{\partial L}{\partial w_{80}} = \frac{\partial L}{\partial z_6} \cdot \frac{\partial z_6}{\partial w_{80}}$$

$$\frac{\partial L}{\partial w_{90}} = \frac{\partial L}{\partial z_6} \cdot \frac{\partial z_6}{\partial w_{90}}$$

To update w of z_1
↳ $\frac{\partial L}{\partial w_{50}} + \frac{\partial L}{\partial w_{51}}$

$$\frac{\partial L}{\partial w_{50}} = \underbrace{\frac{\partial L}{\partial z_6} \cdot \frac{\partial z_6}{\partial z_4} \cdot \frac{\partial z_4}{\partial w_{50}}}$$ → Chain Rule (Piche sai).

$\frac{\partial L}{\partial w_{50}} + \frac{\partial L}{\partial w_{51}}$ = up/slope of z_1

$$\frac{\partial L}{\partial w_{51}} = \underbrace{\frac{\partial L}{\partial z_6} \cdot \frac{\partial z_6}{\partial z_5} \cdot \frac{\partial z_5}{\partial w_{51}}}$$ → Chain Rule (Piche sai)

$$\frac{\partial L}{\partial w_{61}} = \frac{\partial L}{\partial z_6} \cdot \frac{\partial z_6}{\partial z_4} \cdot \frac{\partial z_4}{\partial w_{61}}$$

$$\frac{\partial L}{\partial w_{71}} = \frac{\partial L}{\partial z_6} \cdot \frac{\partial z_6}{\partial z_5} \cdot \frac{\partial z_5}{\partial w_{71}}$$

$$\cancel{\frac{\partial L}{\partial w_{10}} = \frac{\partial L}{\partial z_6} \cdot \frac{\partial z_6}{\partial z_4} \cdot \frac{\partial z_4}{\partial z_1} \cdot \frac{\partial z_1}{\partial w_{10}}}$$

$$\cancel{\frac{\partial L}{\partial w_{11}} = \frac{\partial L}{\partial z_6} \cdot \frac{\partial z_6}{\partial z_6}}$$

$v_1 + v_2 + v_3 \ldots v_n$ (n = no. of paths)

from z_1 there is 2 paths

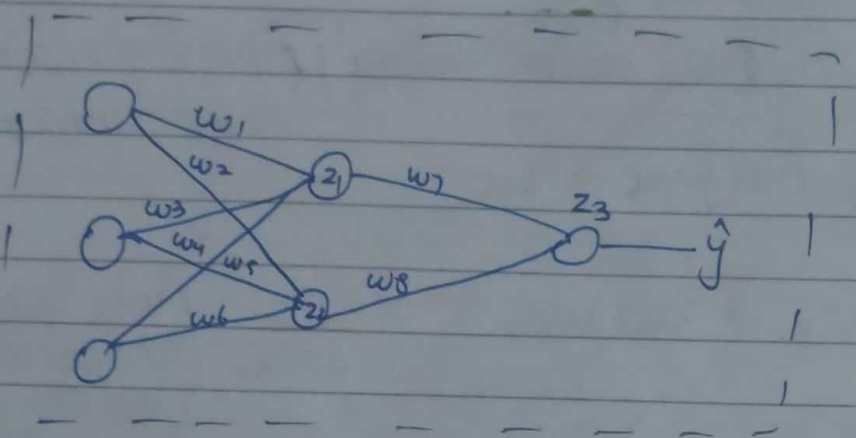
$\frac{\partial L}{\partial z_6} \cdot \frac{\partial z_6}{\partial}$

Chain Rule → affect + kitne path.

$w_{old} = w_{new}$ = No change in movement

Vanishing Gradient Problem

- Occurs (may) using Sigmoid AF. (z → due to Sigmoid AF)

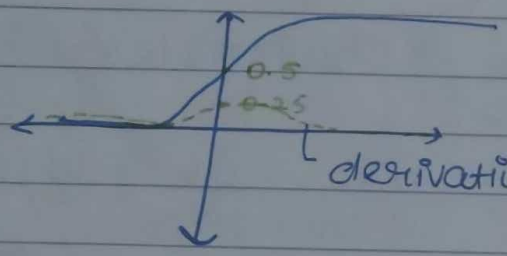


$$w_{1,new} = w_1 - \alpha \frac{\partial L}{\partial w_1} \quad (2)$$

(3) → That's why Don't use Sigmoid AF in all Hidden layers. $\boxed{\text{Result}}$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial z_3} \cdot \frac{\partial z_3}{\partial z_1} \cdot \frac{\partial z_1}{\partial w_1} \quad (1)$$

★ Derivate of Sigmoid → Blw 0 and 0.25



derivative of sigmoid func^n. (0 ⇔ 0.25)

- From (1) → The product we get will get smaller and the further we Backtrack the smaller it will get.

+ Product with α may make it more smaller so in (2)

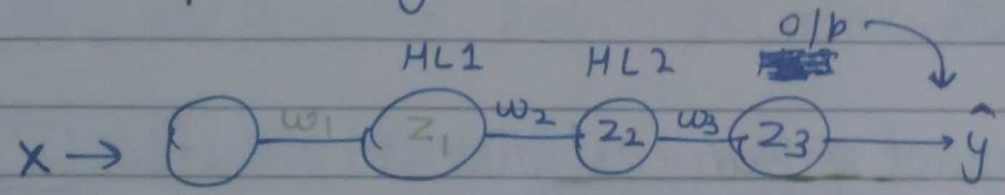
(3) → So $\boxed{w_{new} = w_{old}}$ No change

→ The Result will not be THAT Diff from the old value (No significance diff in new and old value).

→ happens Because of weights.

$O_{21} = z_2$
$O_{11} = z_1$

Exploding Gradient Problem



→ $w_1 = w_1 - \alpha \frac{\partial L}{\partial w_1}$ ①

$$\left[\frac{\partial L}{\partial w_1} = \frac{\partial L}{z_3} \cdot \underbrace{\frac{\partial z_3}{\partial z_2}}_{200} \cdot \frac{\partial z_2}{\partial z_1} \cdot \underbrace{\frac{\partial z_1}{\partial w_1}}_{125 \cdot 100 = 12500} \right]$$

- $z_2 = w_2 \cdot z_1 + b$ → $z_2 = Act(z)$
  $z_2 = \phi(z)$
  sigmoid (0-1)

$$\frac{\partial z_2}{\partial z_1} = \overbrace{\frac{\partial \phi(z_2)}{\partial z_2}}^{0 \leftrightarrow 0.25} \cdot \left[\frac{\partial z_2}{\partial z_1} \right] \rightarrow \frac{\partial (w_2 \cdot z_1 + b)}{\partial z_1}$$

Const

↳ $0 \leq \partial \phi(z_2) \leq 0.25 * w_2$.

↳ $0.25 * w_2 \rightarrow 0.25 \times 500 = 125$

↳ larger value of derivative. Because of weights

↳ old - large value = negative value (-ve)

new value

- Old and new value will vary a lot → GD will never converge

1) Fix → Overfitting

Multi-layer
→ No underfitting
→ But overfitting

DROP-OUT layers in Multi-Neural Networks.
And Regularization.

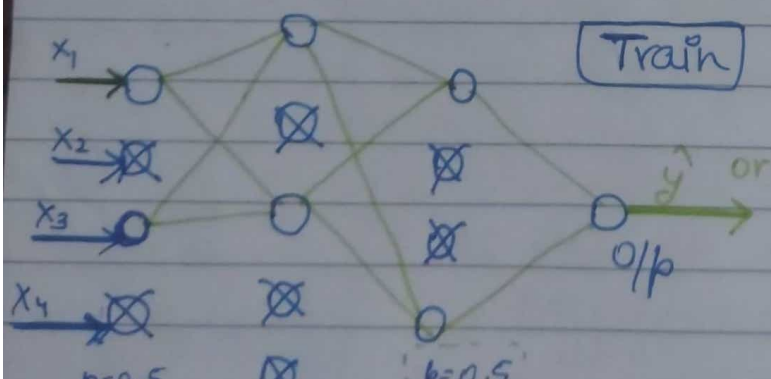
•→ Similar to 'subset of features' given to Multiple DTs in a Random Forest.

1) Select Drop-Out Ratio → (p) → $0 \leq p \leq 1$

2) Select subset of features from I/p layer.
↳ Subset of features / activation func neurons from Hidden Layer
↳ selecting activated neurons from Hidden Layer.

→ Depending on Drop-out Ratio some nodes and neurons are activated and deactivated. In Backtracking, weights of activated neurons are updated, in next iteration randomly some nodes and neurons are selected / activated and deactivated and cycle repeats itself.

For test data - There is no random selection, ✓ everything is connected But each and every weights are multiplied by drop-out Ratio (p) which was selected at the time of training.



Test
↳ everything is connected But
↳ $w = \boxed{w \times p}$.
every weights

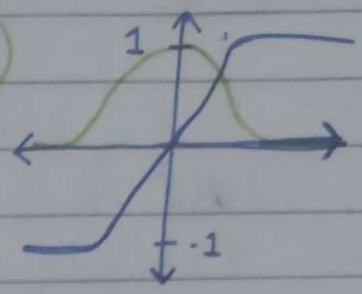
'on this Basis Nodes are activated and deactivated.'

solves Vanishing / Exploading GD problem.

ReLu and Leaky ReLu

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

→ Threshold Activation funcn → f_{th} (-1 to 1)

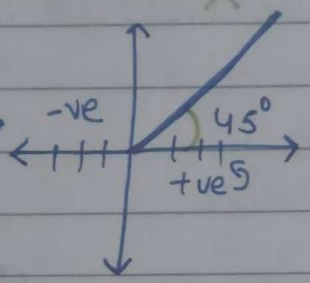


→ $\frac{\partial(f_{th})}{\partial z}$ = (0 to 1) → $\boxed{<1}$

└ give prob to Vanishing and Exploding GD problem.

—x—x—x—x—

•→ Relu → max(y, 0) →



→ tan 45° = 1

→ $\boxed{f_R}$

$\begin{cases} z & z>0 \\ 0 & z<0 \end{cases}$ or max(z, 0)

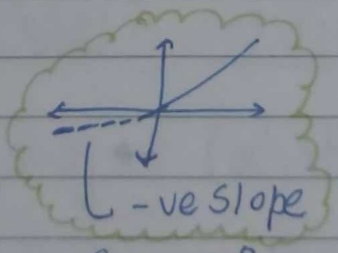
→ Derivative of Relu Act (in Backtracking)

$$\frac{\partial f_R}{\partial z} \rightarrow \begin{cases} 1, & z>0 \\ 0, & z<0 \end{cases}$$

only 2 values i.e., 0 or 1

!! Prob - ? → if one of the derivative Become zero, then again vanishing GD problem will arise

↓ Fix by 'Leaky ReLu'



•→ Leaky Relu (f_{LR}) → when z is negative, do not keep it zero, add some values. (small value addition)

$$f_{LR} = \begin{cases} z, & z>0 \\ 0.01(z), & z<0 \end{cases}$$

negative slope

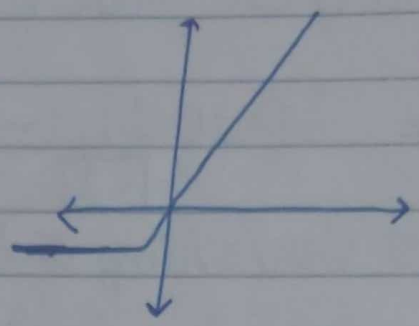
$$\frac{\partial f_{LR}}{\partial z} \rightarrow \begin{cases} 1, & z>0 \\ 0.01, & z<0 \end{cases}$$

-ve slope But not 0.

More Act functions ($x=z$)

→ 1) ELU → (exponential Linear Unit) func^n

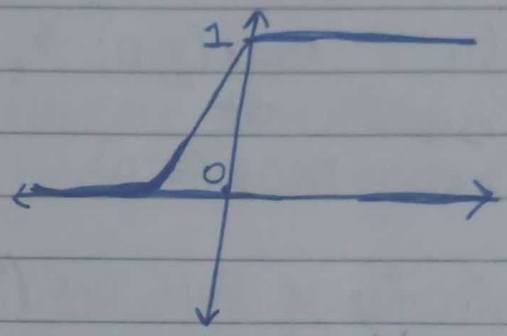
$$= \begin{cases} x, & \text{if } x > 0 \\ \alpha(e^x - 1) & \text{otherwise} \end{cases}$$



→ (f_e)

* Don't have to find derivative of zero.

$$\frac{\partial f_e}{\partial R} \rightarrow \begin{cases} 1. & x > 0 \\ \alpha(e^x) & \text{otherwise} \end{cases}$$



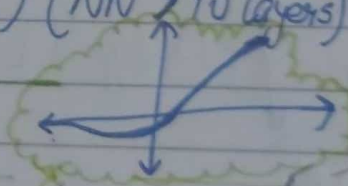
- Because of e^x → Time ↑↑

2) PRelu (Parametric Relu) → (f_{PR})

→ $\begin{cases} x & x > 0 \\ \alpha x & \text{otherwise} \end{cases}$ | if $\alpha = 0.01$ = LRelu, $\alpha = 0$ = Relu | $\frac{\partial \sigma x}{\partial x}$ → $\boxed{\sigma}$ NON ZERO, solves problem

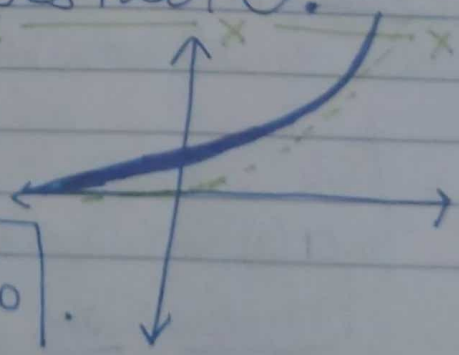
3) Swish Function (Use in LSTM) (NN > 40 layers).

$y = x \cdot \text{sigmoid}(x) \rightarrow y = x * \sigma(x)$.



→ Solves dead Relu as it passes near 0.

4) $\boxed{\text{Softplus}}$ →



$y = \ln(1 + e^x)$ (No need of der of 0).

works on probab and real values

5) Softmax → $S(x_j) = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}$, $j = 1, 2, 3, \ldots K$.

→ for an arbitary real vector of length K, softmax can compress it into real vector of length K with a value in the Range (0, 1) and the sum of the elements in the vector is 1

ex 1

→ Before the Input to the O/p layer or I/p of o/p layer

$\{x_1, x_2, x_3, x_4\}$

$$\frac{e^{x_1}}{e^{x_1}+e^{x_2}+e^{x_3}+e^{x_4}} + \ldots + \frac{e^{x_4}}{e^{x_1}+e^{x_2}+e^{x_3}+e^{x_4}} \rightarrow \{P_1, P_2, P_3, P_4\}$$

highest P → Ans.

O/p

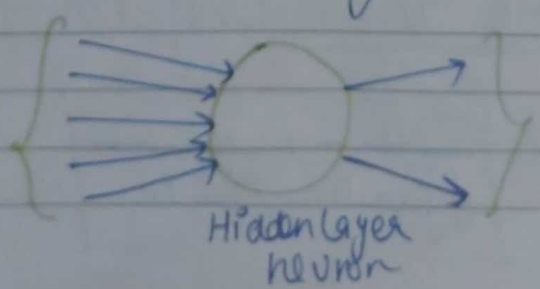
* Sigmoid and Softmax act fnch are kept at last layer.
(>0.5, <0.5) (Range of Probab)

—X—X—X—X—

* WEIGHT INTIALIZATION

1) weight should be small.
2) weight should not be same.
3) weight should have good Variance.

Research Terms →



METHOD-1: Uniform Distribution

$$W_{ij} \sim \text{uniform}\left[\overset{a}{\frac{-1}{\sqrt{fan\_in}}}, \overset{b}{\frac{1}{\sqrt{fan\_in}}}\right]$$

→ Good with Sigmoid Activation Function

METHOD-2: Xavier/Gorat Normal Distribution

$$W_{ij} \sim N(0, \sigma) \quad (mean=0 \mid sd=\sigma)$$

$$\sigma = \sqrt{\frac{2}{(fan\_in + fan\_out)}}$$

METHOD-2.1 → Xavior/Gorat Uniform Distribution

$$W_{ij} \sim U\left[\overset{a}{\frac{-\sqrt{6}}{\sqrt{fan\_in + fan\_out}}}, \overset{b}{\frac{\sqrt{6}}{\sqrt{fan\_in + fan\_out}}}\right]$$

→ good with Sigmoid Activation Function

METHOD-3: He init uniform Distribution.

$$W_{ij} \sim U\left[\overset{a}{\frac{-\sqrt{6}}{\sqrt{fan\_in}}}, \overset{b}{\frac{\sqrt{6}}{\sqrt{fan\_in}}}\right]$$

GD → All points → Loss $= \frac{1}{2} \sum_{i=1}^{n} (y-\hat{y})^2$

SGD → one by one → loss $= \frac{(y-\hat{y})^2}{2}$

METHOD 3.1 → He init Normal Distribution

$$W_{ij} \sim N(0, \sigma)$$

$$\sigma = \sqrt{\frac{2}{fan\text{-}in}}$$

→ good for Relu, LRelu, elu, P-Relu and others.

---

* STOCHASTIC GD [SGD]

→ Find error → find partial derivative → finding slope → fixing weights

↳ one by one → $\boxed{SGD}$. [iterations / epocs will ↓) F+Backward]

$$W_{new} = W_{old} - \alpha \frac{dL}{dW_{old}} \rightarrow \frac{dL}{dW_{old}}$$

GD → Loss(L) → $\sum_{i=1}^{n} (y-\hat{y})^2$

SGD → loss(L) → $(y-\hat{y})^2$ → like in L.R

m.B.SGD = loss(L) → $\sum_{i=1}^{K} (y-\hat{y})^2$

↳ n data points (all) → GD

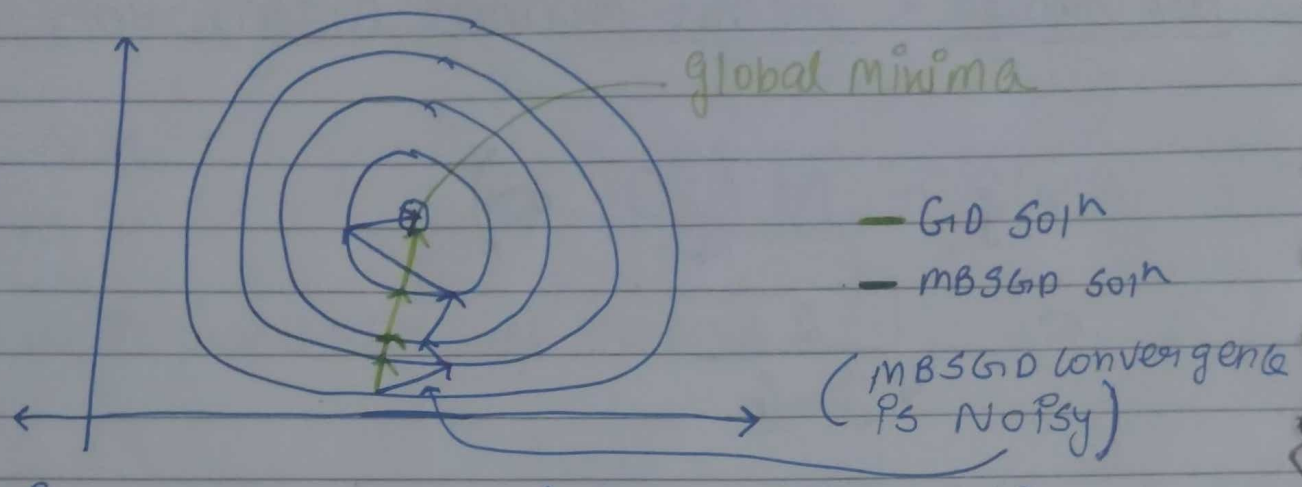
Req more comp. Power To Load

↳ 1 data Point → SGD

very slow

↳ K data points (K<n) → mini Batch SGD

3D - 2D representation of GD and MB SGD



The zig-zag movement of MBSGD Sol^n is known as **NOISY DATA**

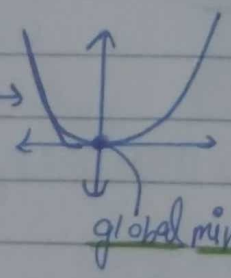
↳ We use MBSGD with Momentum to eliminate this or remove this. (Noise).

$$\left[\frac{\partial L}{\partial w_{old}}\right]_{MSSGD} \approx \left[\frac{\partial L}{\partial w_{old}}\right]_{GD}$$

↳ sample ↳ population

---

* Global and local minima

1) Not every loss func^n gives GD graph → 

$(y-\hat{y})^2$

2) some loss func^n may give graph ↳