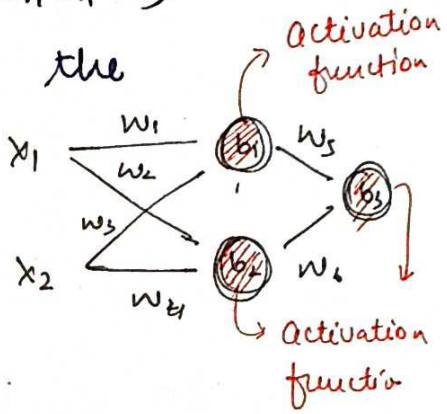
Activation Function

What are Activation functions?

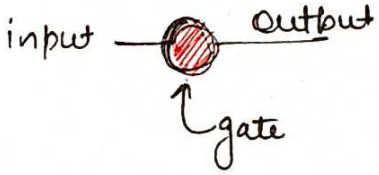
In artificial neural Networks, each neuron forms a weighted sum of its inputs and passes the resulting scalar value through a function referred to as an activation then the output or activation of a neuron is

$$a = g(w_1x_1 + w_2x_2 + w_3x_3 + \dots + w_nx_n + b)$$

The function g is referred to as the activation function.

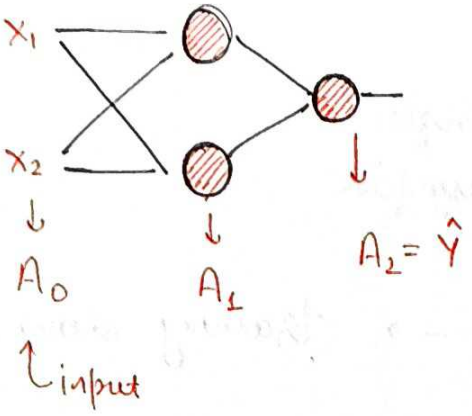


basically, Activation function is a mathematically gate



If the function g is taken as the linear regression function $g(z) = z$ then the neuron perform linear regression or classification. In general g is taken to be a non-linear function. If the function g to do non-linear regression and solve classification problems that are not linearly separable.

Why activation functions are needed?



Forward propagation

$$z_1 = w_1 A_0 + b_1$$

$$A_1 = g(z_1) = z_1$$

We are not using any activation function, so our output is same (z_1).

$$A_2 = g(w_2 A_1 + b_2)$$

$$= w_2 A_1 + b_1 = w_2 (w_1 A_0 + b_1) + b_2$$

$$= w_2 w_1 A_0 + w_2 b_1 + b_2$$

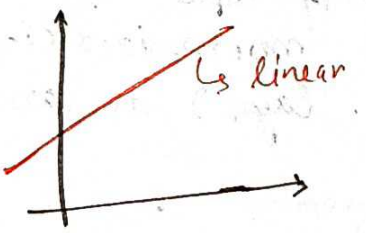
$$= w' A^0 + b'$$

output

input

$$A_2 = \hat{y} = w' A_0 + b'$$

↳ one degree polynomial



Ideal Activation function

1.) Non-linear → Good activation functions are non-linear.

$y = f(x)$ → relation betn y and x should be non-linear

for eg:- Sigmoid → $\sigma(z) = \frac{1}{1+e^{-z}}$

* Universal Approx Theorem :→ With the help of enough enough node and layer we can capture any type of non-linear pattern.

2) Differentiable → Activation function should be differ-tiable

3) Computationally inexpensive → derivative → Singles easy fast

↳ If computation expensive → training slow

4) Zero centered → Output of activation function should be

- Zero center
  - mean = 0
  - example → tanh
- Normalize
  - Neural Network fastly converge if input data is normalized (input layer) or later layer

5] Non-Saturating

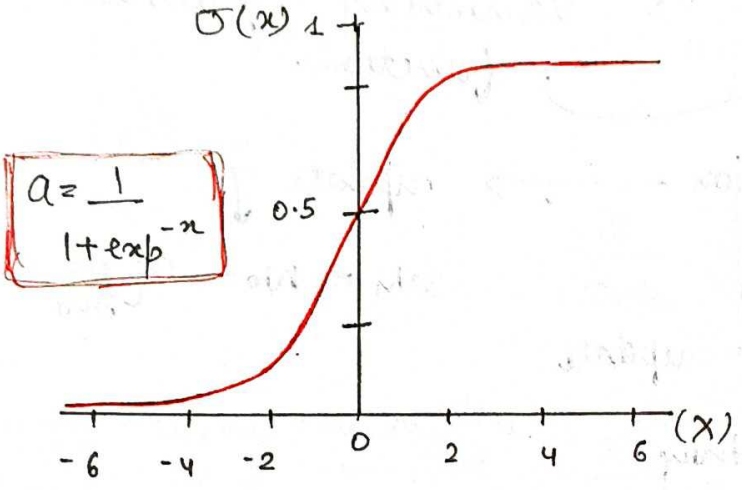
Saturating means to squeeze the input in particular range. For eg:- Sigmoid → [0, 1], tanh → [-1, 1]

Saturating function face → Vanishing Gradien problem

Non-Saturating → Relu → (0, x)

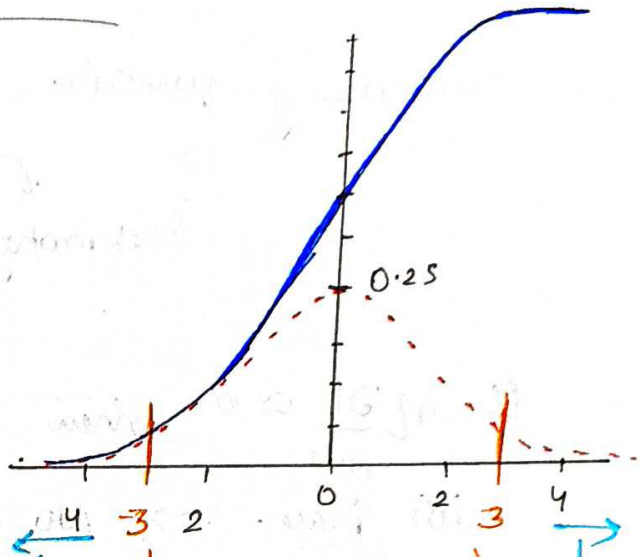
↳ No range

Sigmoid Activation function



$a = \frac{1}{1+\exp^{-x}}$

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



↳ derivative value is good

near to zero
$\sigma(x) = 0$

Advantages

1.) Output of sigmoid is bet^n (0, 1)

↳ So, we can treat as a probability and use at output layer.

use in Binary classification ←

2.) Non-linear → If your data is Non-linear So, sigmoid function can capture Non-linearity. (good option).

3.) Differentiable → Backpropagation

↓

$\frac{dL}{dw}$

Disadvantage

1) Saturating function → facing → Vanishing gradient function.

Backpropagation → update

$$W_n = W_o - \eta \frac{\partial L}{\partial w}$$

if $\eta \frac{\partial L}{\partial w} \simeq 0$ then No update will place. → No Training

when $\eta \frac{\partial L}{\partial w}$ will be 0? → $x \uparrow\uparrow$ (large)

$w_1 x_1 + w_2 x_2 + b$

↳ if w value is large then x value is also large

when we go to other layers Sigmoid output will be 0. (Layer ke piche jate jayega output chhoti hoti jayegi fir ek time pe 0 ho jayega.)

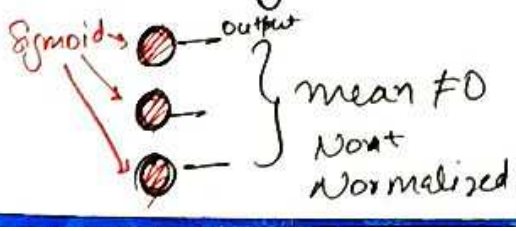
New weight = = Old weight.

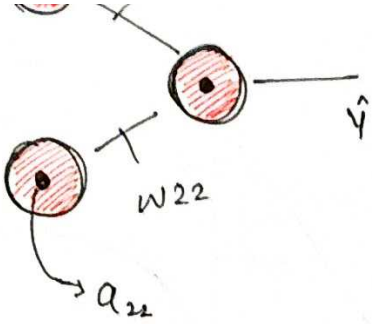
That's why Sigmoid not use in hidden layer only use in outer layer.

2) Non-Zero centered

After applying activation function (Sigmoid) all values is not normalized and mean ≠ 0.

Training is slow bcz of not normalized and mean ≠0





$\sigma(z)$ ↵ w_{21} a_{22} w_{22}

$$\frac{\partial L}{\partial w_{21}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_{31}} \frac{\partial z_{31}}{\partial w_{21}}$$

Same ↕

$$\frac{\partial z_{31}}{\partial w_{21}} = a_{21} w_{21} + a_{22} w_{22} + b_{31}$$

$a_{21} w_{21}$ → 1 ; $a_{22} w_{22} + b_{31}$ → 0

$$\frac{\partial L}{\partial w_{22}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_{31}} \frac{\partial z_{31}}{\partial w_{22}}$$

$$\frac{\partial z_{31}}{\partial w_{21}} = a_{21}$$

$$\frac{\partial z_{31}}{\partial w_{22}} = a_{22}$$

→ Output of Sigmoid (0→1) → +ve

① $\frac{\partial L}{\partial w_{21}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_{31}} a_{21}$ → +ve

② $\frac{\partial L}{\partial w_{22}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_{31}} a_{22}$ → ve

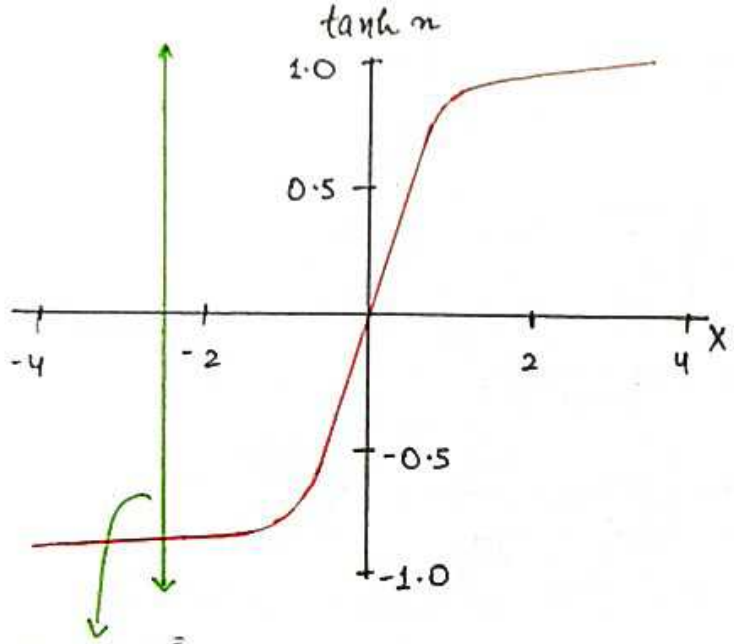
If ① is (-ve) then ② also (-ve)
If ① is positive then ② also (+ve)
Cz both are same.

Weight's derivation sare (+ve) hoyge
ya sare (-ve) hoyge.
(1 positive or 1-ve) nhi hoyge.

That's mean this is restricted

3) Sigmoid formula → $\frac{1}{1+e^{-x}}$ → computationally expensive

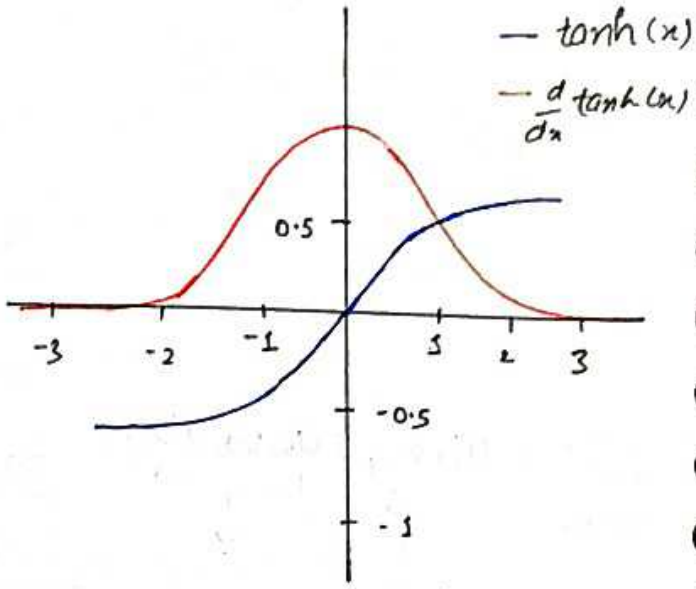
Tanh Activation function



[-1, 1]

formula]

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



↳ after derivative

$$\text{ders} = (1 - \tanh^2(x))$$

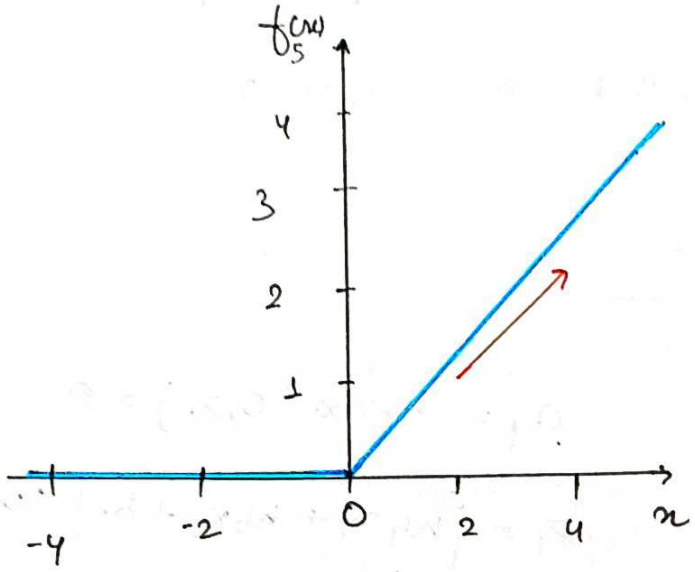
Advantage

1) Non-linear
2) Differentiable
3) Zero-centered ⇝ gradient is (+ve) and (-ve) too
   ↳ training faster than Sigmoid

Disadvantage

1. Saturating function → vanishing gradient problem
2. Computational expensive

Relu Activation function



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

Advantage:

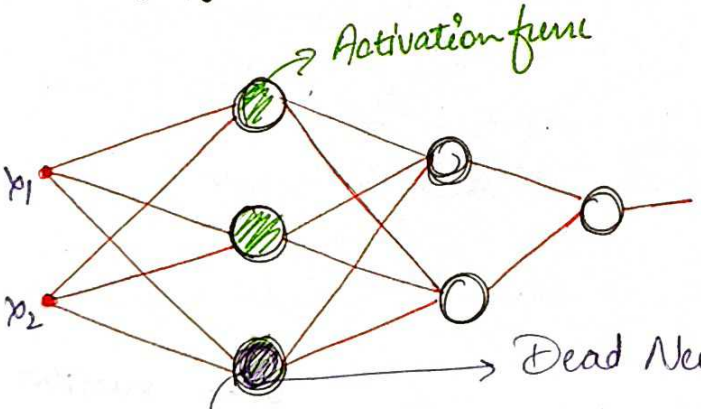
1) Non-linear
2) Not Saturated in the positive region.
   Positive region kitna bhi aage jao sikta hai. No limit
3) Computationaly Inexpensive
4) Convergence → faster than Sigmoid and tanh.

Disadvantage:

1) Differentiate → not zero.
   So assumption for this
   $x < 0 \rightarrow 0$
   $x \geq 0 \rightarrow 1$

2) Not zero centered

Dying Relu Problem



Dead Neuron → forever dead

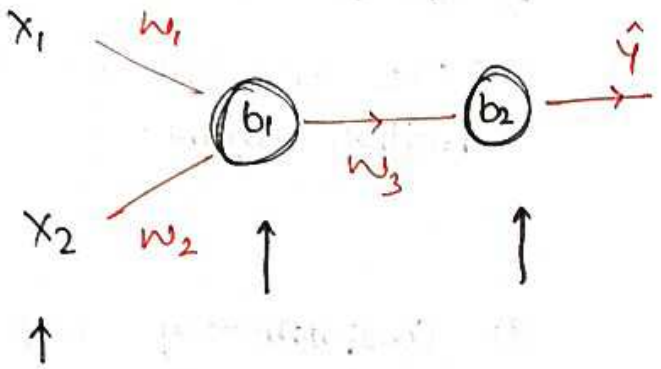
if one time Neuron is dead then everytime neuron is dead

Some Neuron's output is 0. And Not depend on input bcz you can give any value still output is 0.

If 50% > or more than 50% of neuron/node are dead then model or architecture cannot capture pattern.

Worst case → 100% of node are dead neurons

Why dying Relu Problem happen?



$$a_1 = \max(0, z_1)$$

$$z_1 = \boxed{w_1 x_1 + w_2 x_2 + b_1}$$

Let assume

$$z_1 = \boxed{w_1 x_1 + w_2 x_2 + b_1} < 0$$

then

$$a_1 = \max(0, z_1) = 0$$

If a_1 is 0 then derivation of a_1 w.r.t z_1 is 0.

$$\boxed{\frac{\partial a_1}{\partial z_1} = 0}$$

→ If a_1 is 0. Then w_1 and w_2 are not update.

$$w_1 = w_1 - \eta \boxed{\frac{\partial L}{\partial w_1}}$$

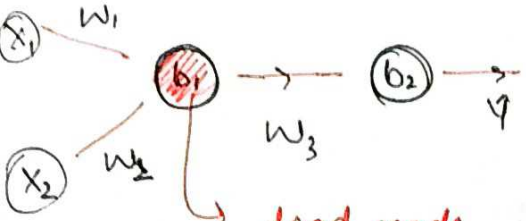
$$w_2 = w_2 - \eta \boxed{\frac{\partial L}{w_2}}$$

Use this term to solve this both derivation.

$$w_1 = w_1 - \eta(0) \longrightarrow w_1 = w_1$$

$$w_2 = w_2 - \eta(0) \longrightarrow w_2 = w_2$$

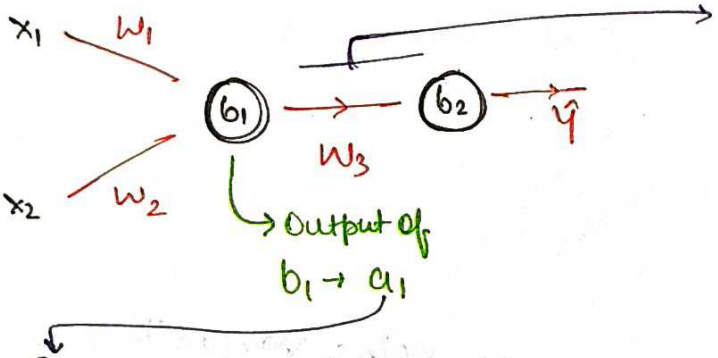
both weight are same. No change in weight



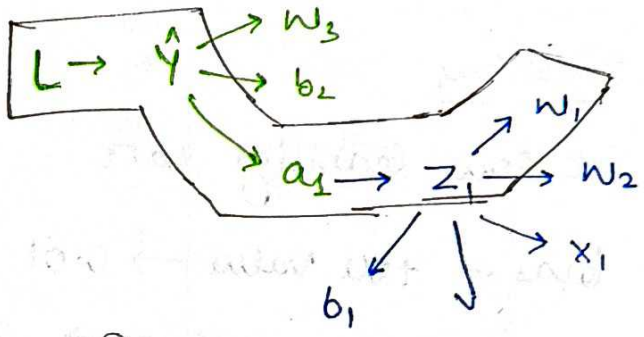
dead node
(w_1 and w_2 always same)

→ Let assume neural network is regression for proove that $\frac{\partial a_1}{\partial z_1}$ → term is use to

Solve $\frac{\partial L}{\partial w_1}$ and $\frac{\partial L}{\partial w_2}$.



Regression Neural Network



$a_1 = \max(0, z_1) = 0$

$z_1 = w_1x_1 + w_2x_2 + b_1$

$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_1} \frac{\partial a_1}{\partial z_1} \frac{\partial z_1}{\partial w_1}$ ← $\frac{\partial a_1}{\partial z_1} = 0$; x_2 → 0

$w_{1\,new} = w_{1\,old} - \eta \frac{\partial L}{\partial w_1}$ ← 0

$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_1} \frac{\partial a_1}{\partial z_1} \frac{\partial z_1}{\partial w_2}$ → 0

$w_{2\,new} = w_{2\,old} - \eta \frac{\partial L}{\partial w_2}$ ← 0

$\frac{\partial L}{\partial w_2}$ → $c_2 \frac{\partial a_1}{\partial z_1}$

$w_{new_1} = w_{old_1}$ → Same

$w_{new_2} = w_{old_2}$ → Same

When is this term $z_1 = w_1x_1 + w_2x_2 + b_1$ negative ?

reason → 1. High learning rate

$$W_1 = W_1 - \eta \frac{\partial L}{\partial W_1}$$

W_1 → Small Number

η → High lr

$\frac{\partial L}{\partial W_1}$ → any number → Very high positive number cz l.r

$W_1 = (-ve)$ → Subtract Small number − large number.

In Next cycle z_1 will be negative cz of W_1 is negative.

2) High Negative Bias

$$z_1 = W_1 x_1 + W_2 x_2 + b_1$$

b_1 → high −ve

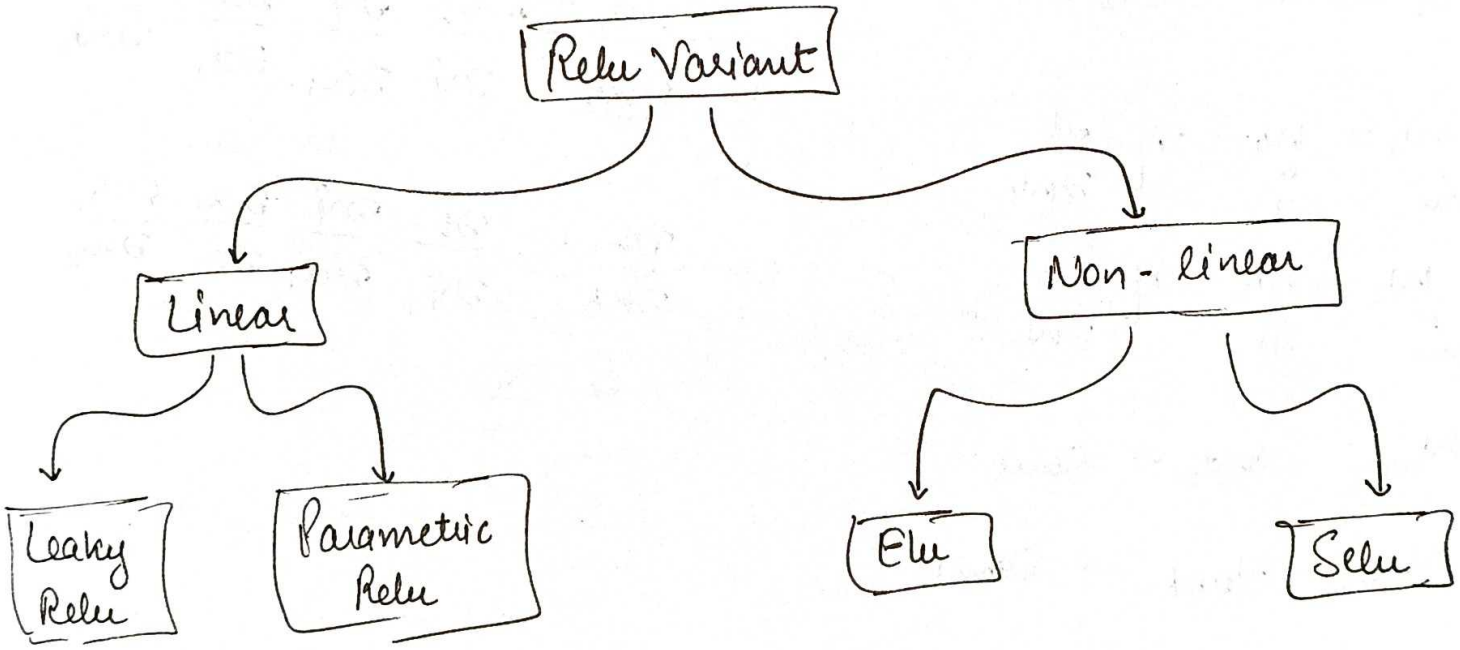
z_1 = also negative

Solutions

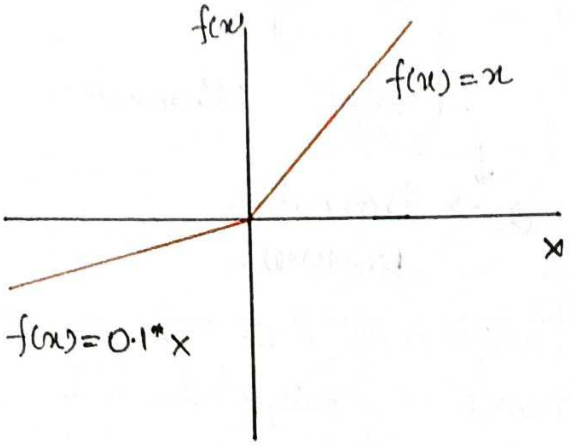
→ Set low learning rate

→ bias → +ve value → 0.01

→ Don't use relu → you can use relu variant.



Leaky Relu



$$f(z) = \max(0.01z, z)$$

$z \geq 0 \rightarrow z$

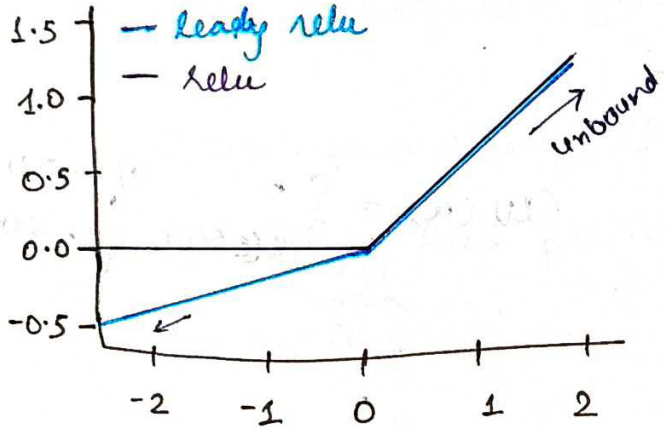
$z \leq 0 \rightarrow \frac{1}{100} z$ (fraction of z)

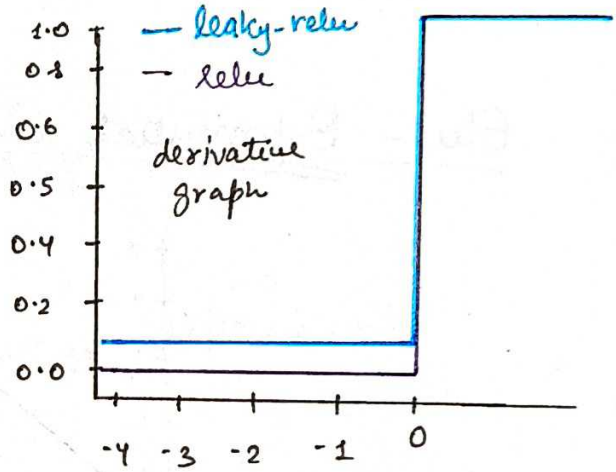
when (z) → -ve then derivative is not 0. its 0.01

So, in system small gradient flow

$$w_1 = w_1 - \eta \left(\frac{\partial L}{\partial w_1} \right)$$ → not 0 / → 0.01

So, not facing any dying Relu problem.





Advantages

(i) Non-saturated (both direction → unbounde (not restriction)

(ii) Easily computed

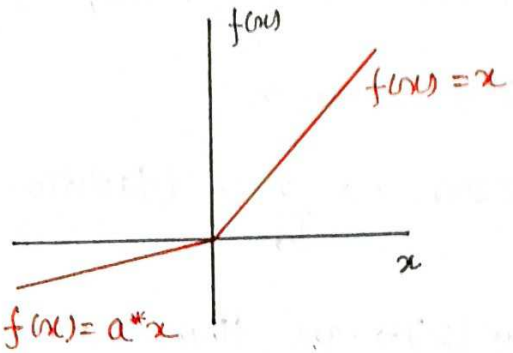
(iii) No dying relu problem

(iv) Close to center → cz both type of value present
-ve +ve

why we take 0.01 in formula?

↳ will discuss in parametric Relu.

Parametric Relu



$$f(x) = \begin{cases} x & \text{if } x > 0 \\ ax & \text{Otherwise} \end{cases}$$

a → trainable parameter

0.01 in Leaky relu

Parametric Relu is exactly same → Leaky relu. Only difference is Leaky relu and parametric relu

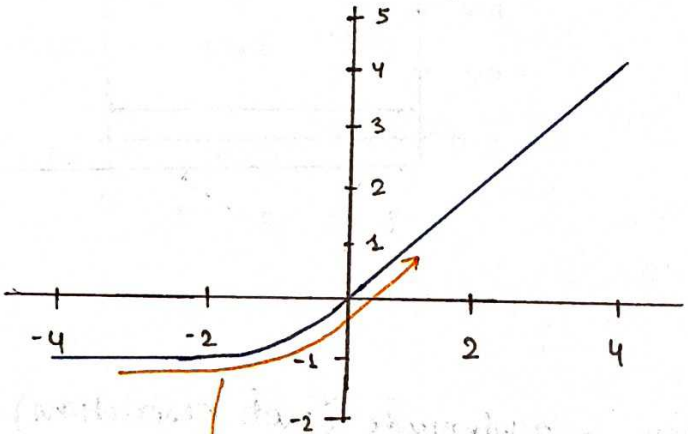
Leaky relu → in negative side $f(x) = 0.01 * x$

parametric relu → in negative side $f(x) = a * x$

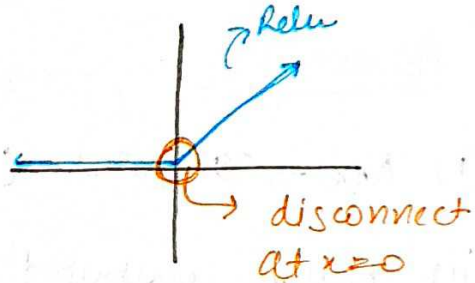
Elu – Exponential Linear Unit

$$ELU(x) = \begin{cases} x & \text{if } x > 0 \\ a(e^x - 1) & \text{if } x < 0 \end{cases}$$

Constant (0.1 to 0.3)



Continous (Not disconnect at x=0) like Relu

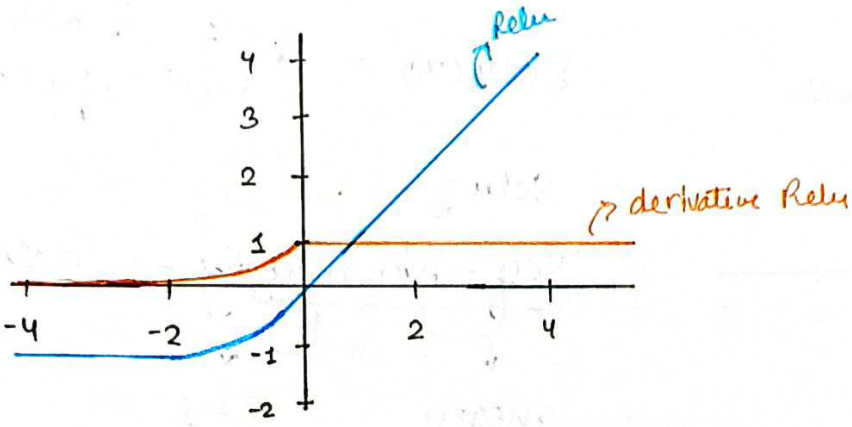


→ Elu →
- always continous
- always differentiable

Differentiable formula of ELU → for positive

$$ELU'(x) = \begin{cases} 1 & \text{if } x > 0 \\ ELU(x) + \alpha & \text{if } x \leq 0 \end{cases}$$

for negative



Advantage

→ Close to zero centred
↳ convergence faster

→ Better generalized (Good in train.)

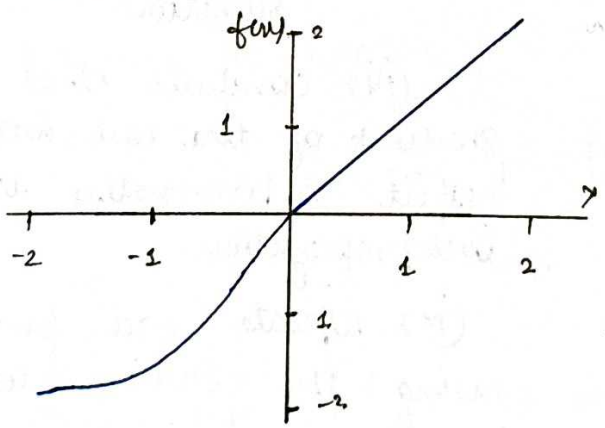
→ No Dying Relu problem

→ Always continous and always differentiable

Disadvantage

→ computation expensive

Selu – Scaled Exponential Linear Unit

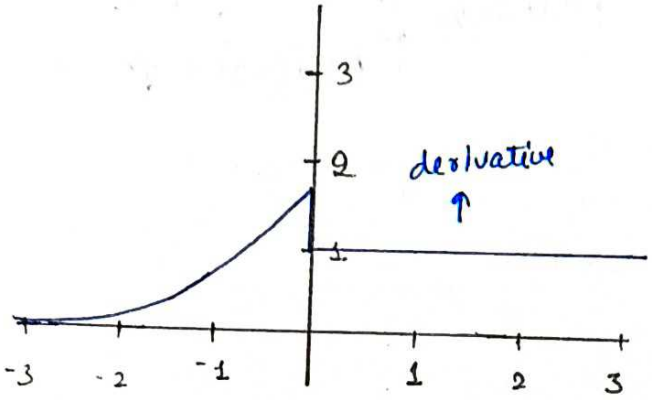


$$SELU(x) = \lambda \begin{cases} x & \text{if } x > 0 \\ \alpha e^x - \alpha & \text{if } x \leq 0 \end{cases}$$

experimental fix parameter

$\alpha = 1.6732632423543772$

$\lambda = 1.0507009873554$



$$SELU'(x) = \lambda \begin{cases} 1 & \text{if } x > 0 \\ \alpha e^x & \text{if } x \leq 0 \end{cases}$$

Selu ↓

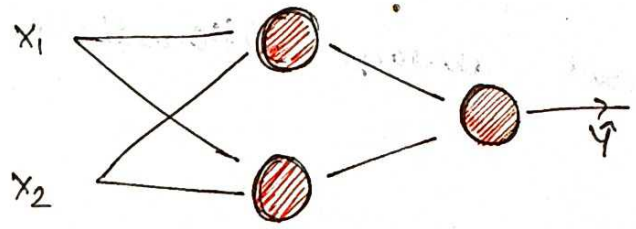
Self-Normalizing → activation

mean 0 $\quad$ $\sigma \geq 1$

convergence faster

Weight Initialization

Why weight initialization is important?



You will face problem if intial parameter is wrong.

→ vanishing gradient problem

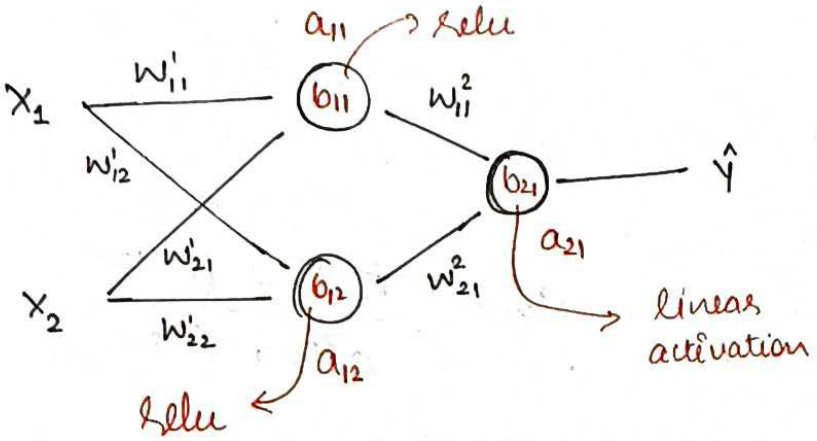
→ Exploding gradient problem

→ slow convergence

1. Initialize the parameter
2. Choose an optimization algorithm.
3. Repeat these steps:
   (i) Forward propagate an input.
   (ii) Compute the cost function.
   (iii) Compute the gradients of the cost with respect to parameter using backpropagation.
   (iv) Update each parameter using the gradients. according to the optimization algo.

What not to do?

Case 1 → Zero Initialization (Regression Example)



Relu activation

$$a_{11} = \max(0, z_{11})$$

$$z_{11} = w'_{11}x_1 + w'_{21}x_2 + b_{11}$$

$$a_{12} = \max(0, z_{12})$$

$$z_{12} = w'_{12}x_1 + w'_{22}x_2 + b_{12}$$

We know, → Initial weight is 0. (z of zero initialization
→ bias is 0.

$w = 0 \quad b = 0$

So,

$$z_{11} = w'_{11}x_1 + w'_{21}x_2 + b_{11} = 0$$

$$a_{11} = \max(0, z_{11}) = 0$$

$$z_{12} = w'_{12}x_1 + w'_{22}x_2 + b_{12} = 0$$

$$a_{12} = \max(0, z_{12}) = 0$$

$a_{11} = a_{12}$ → equal and zero

during propagation

$$w'_{11} = w'_{11} - \eta \frac{\partial L}{\partial w'_{11}}$$

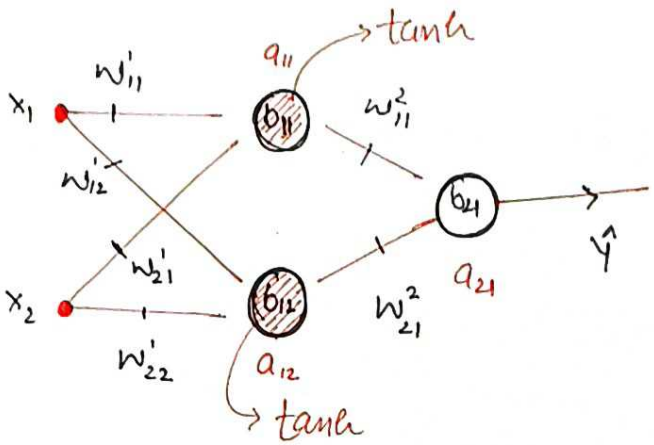
→ for solving this, we also need a_{11} derivative but a_{11} is 0 then this term is 0.

w'_{11} new = w'_{11} old

→ No change or update

→ No training

* In Relu activation → Not initialize 0 in weight during starting.



tanh activation

$$a_{11} = \frac{e^{z_{11}} - e^{-z_{11}}}{e^{z_{11}} + e^{-z_{11}}}$$

$$a_{12} = \frac{e^{z_{12}} - e^{-z_{12}}}{e^{z_{12}} + e^{z_{12}}}$$

$$z_{11} = w'_{11}x_1 + w'_{21}x_2 + b_{11}$$

$$z_{12} = w'_{12}x_1 + w'_{22}x_2 + b_{12}$$

We know that, Initial weight and bias is 0. → zero initialization

$$w = 0, \quad b = 0$$

So,

$$z_{11} = w'_{11}x_1 + w'_{21}x_2 + b_{11} = 0$$

$$a_{11} = \frac{e^{z_{11}} - e^{-z_{11}}}{e^{z_{11}} + e^{-z_{11}}} = 0$$

$$z_{12} = w'_{12}x_1 + w'_{22} + b_{12} = 0$$

$$a_{12} = \frac{e^{z_{12}} - e^{-z_{12}}}{e^{z_{12}} + e^{z_{12}}} = 0$$

$a_{11} = a_{12}$ → equal and zero

during parapagation

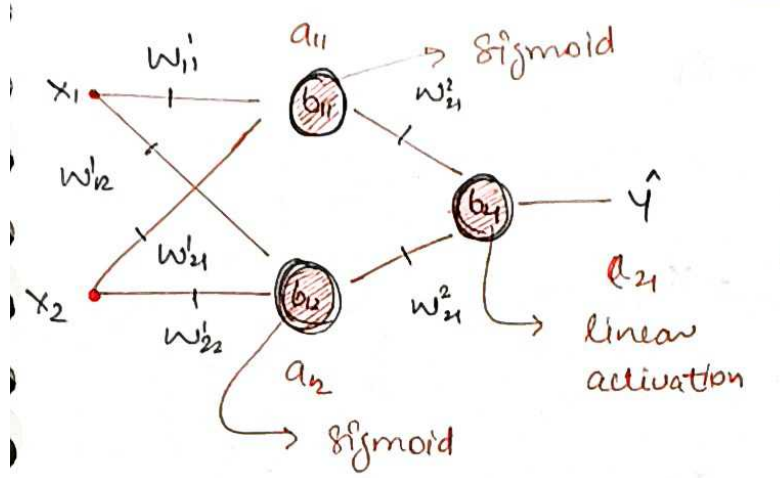
$$w'_{11} = w'_{11} - \eta \frac{\partial L}{\partial w}$$

→ to solving this, we also need a_{11} derivatives but a_{11} is 0. then this term is 0.

w'_{11} (New) = w'_{11} (old) → No update

→ No training

→ In tanh activation → Not initialize is 0 in weight and bias during starting.



Sigmoid Activation

$$a_{11} = \sigma(z_{11})$$

$$z_{11} = w'_{11}x_1 + w'_{21}x_2 + b_{21}$$

$$a_{12} = \sigma(z_{12})$$

$$z_{12} = w'_{12}x_1 + w'_{22}x_2 + b_{12}$$

We know that, initial weight and bias is 0.

$$w = 0 \quad , b = 0$$

So,

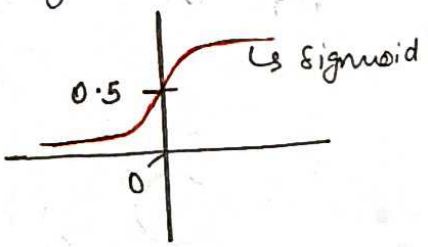
$$z_{11} = w'_{11}x_1 + w'_{21}x_2 + b_{11} = 0$$

$$a_{11} = \sigma(z_{11}) = 0.5$$

$$z_{12} = w'_{11}x_1 + w'_{21}x_2 + b_{11} = 0$$

$$a_{12} = \sigma(z_{12}) = 0.5$$

a_{11} and a_{12} is 0.5 because in sigmoid at 0 is 0.5



$$\boxed{a_{11} = a_{12}}$$

$$w = w - \eta \boxed{\frac{\partial L}{\partial w}}$$ Solving this term for every weight

$$\frac{\partial L}{\partial w'_{11}} = \boxed{\frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_{11}}} \boxed{\frac{\partial a_{11}}{\partial z_{11}} \frac{\partial z_{11}}{\partial w'_{11}}}$$

$$\frac{\partial L}{\partial w'_{12}} = \boxed{\frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_{12}}} \boxed{\frac{\partial a_{12}}{\partial z_{12}} \frac{\partial z_{12}}{\partial w'_{12}}}$$

$$\frac{\partial L}{\partial w'_{21}} = \boxed{\frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_{11}}} \boxed{\frac{\partial a_{11}}{\partial z_{11}} \frac{\partial z_{11}}{\partial w'_{21}}}$$

$$\frac{\partial L}{\partial w'_{22}} = \boxed{\frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_{12}}} \boxed{\frac{\partial a_{12}}{\partial z_{12}} \frac{\partial z_{12}}{\partial w'_{22}}}$$

$a_{11} = a_{12}$
$z_{11} = z_{12}$

$$\frac{\partial z_{11}}{\partial w_{11}'} = \underbrace{w_{11}' x_1}_{1} + \underbrace{w_{21}' x_2 + b_{11}}_{0} = x_1$$

$$\frac{\partial z_{12}}{\partial w_{12}'} = \underbrace{w_{12}' x_1}_{1} + \underbrace{w_{22}' x_2 + b_{12}}_{0} = x_1$$

$$\frac{\partial z_{11}}{\partial w_{21}'} = \underbrace{w_{11}' x_1}_{0} + \underbrace{w_{21}' x_2}_{1} + \underbrace{b_{11}}_{0} = x_2$$

$$\frac{\partial z_{12}}{\partial w_{22}'} = \underbrace{w_{12}' x_1}_{0} + \underbrace{w_{22}' x_2}_{1} + \underbrace{b_{12}}_{0} = x_2$$

Same

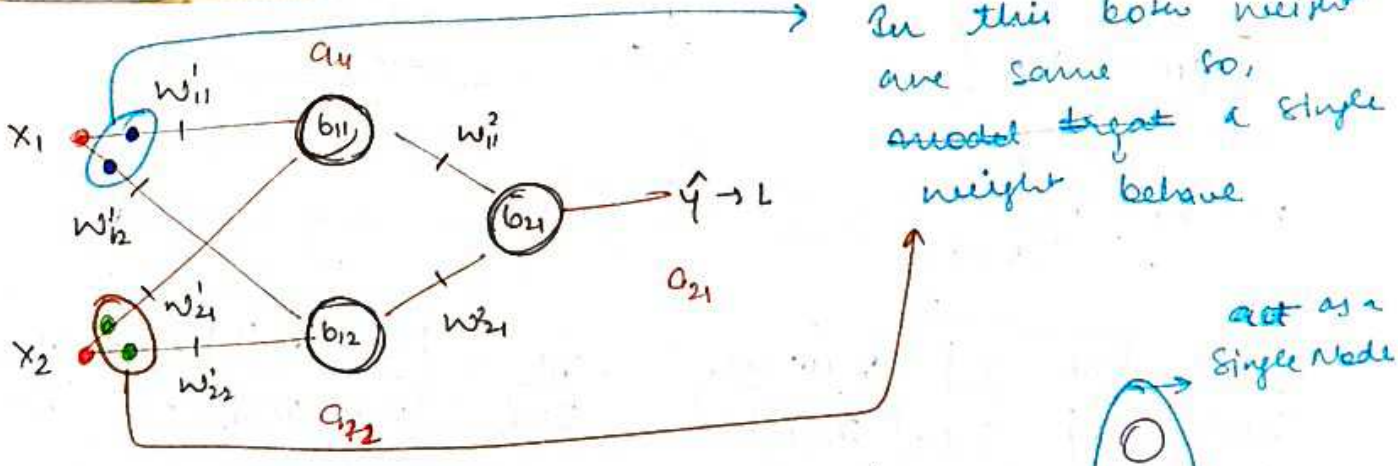
$$\frac{\partial L}{\partial w_{11}'} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_{11}} \frac{\partial a_{11}}{\partial z_{11}} x_1$$

$$\frac{\partial L}{\partial w_{12}'} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_{12}} \frac{\partial a_{12}}{\partial z_{12}} x_1$$

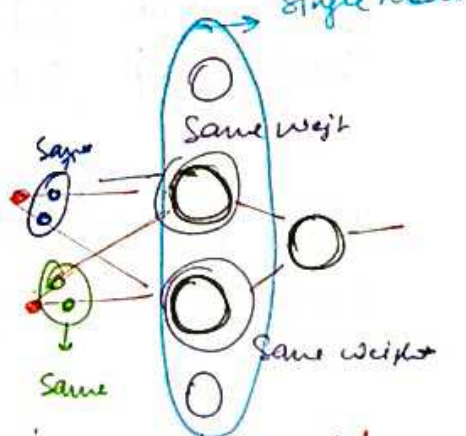
Same

$$\frac{\partial L}{\partial w_{21}'} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_{11}} \frac{\partial a_{11}}{\partial z_{11}} x_2$$

$$\frac{\partial L}{\partial w_{22}'} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_{12}} \frac{\partial a_{12}}{\partial z_{12}} x_2$$

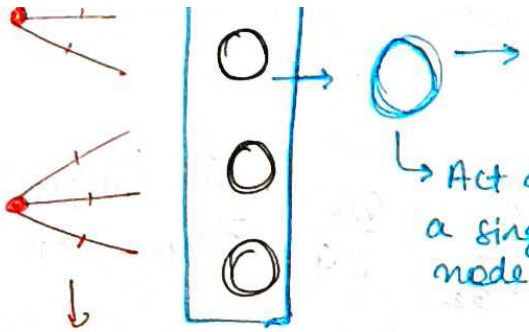


In this both weight are same so, ~~model~~ ~~input~~ a single weight behave

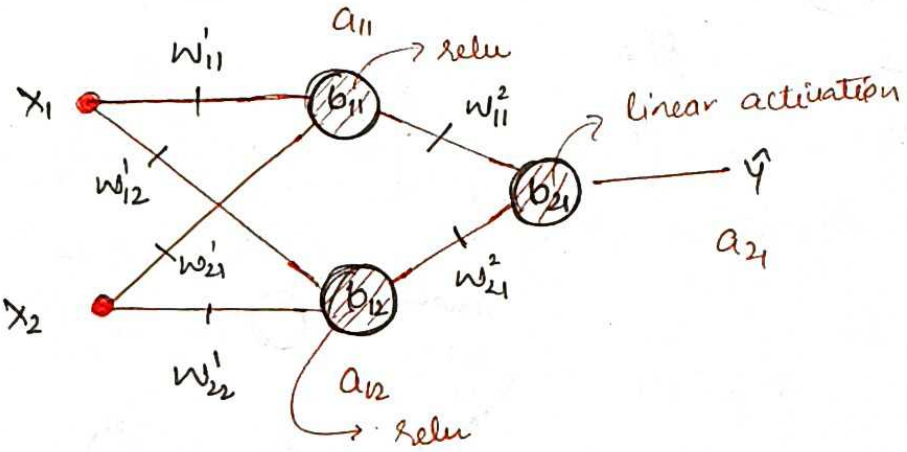


Insight → If we use sigmoid and initialize (w) with 0. So, even we add 100 or 1000 node but it will act as a Single Node.

Every Node (Hidden) contain Same Value (input). and this model convert into Linear model



Case 2 → Non-0 constant value



let $w = 0.5$, $b = 0.5$

$a_{11} = \max(0, z_{11})$

$z_{11} = w'_{11} x_1 + w'_{21} x_2 + b_{11}$

$z_{11} \neq 0$ (Non-zero value)

$a_{11} = \max(0, z_{11}) \neq 0$

$a_{12} = \max(0, z_{12})$

$z_{12} = w'_{12} x_1 + w'_{22} x_2 + b_{12}$

$z_{12} \neq 0$ (Non-zero)

$a_{12} \neq 0$

Initial weight and bias are same (0.5) in z_{11} and z_{12}. So,

$$\boxed{z_{11} = z_{12}} \longrightarrow \boxed{a_{11} = a_{12}}$$

$$\frac{\partial L}{\partial w'_{11}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_{11}} \frac{\partial a_{11}}{\partial z_{11}} x_1$$

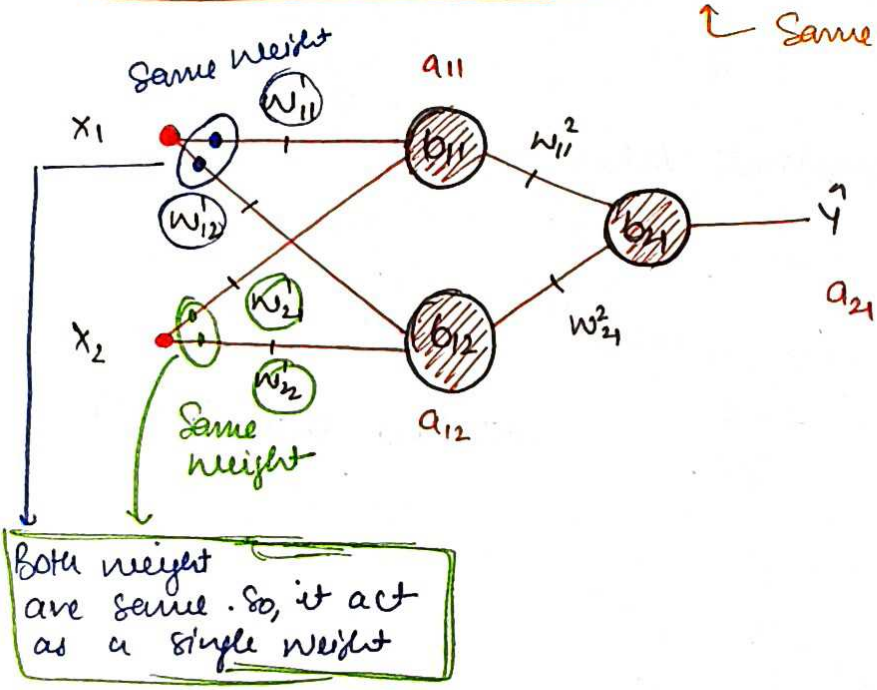
$$\frac{\partial L}{\partial w'_{12}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_{12}} \frac{\partial a_{12}}{\partial z_{12}} x_1$$

↑ Same ↑

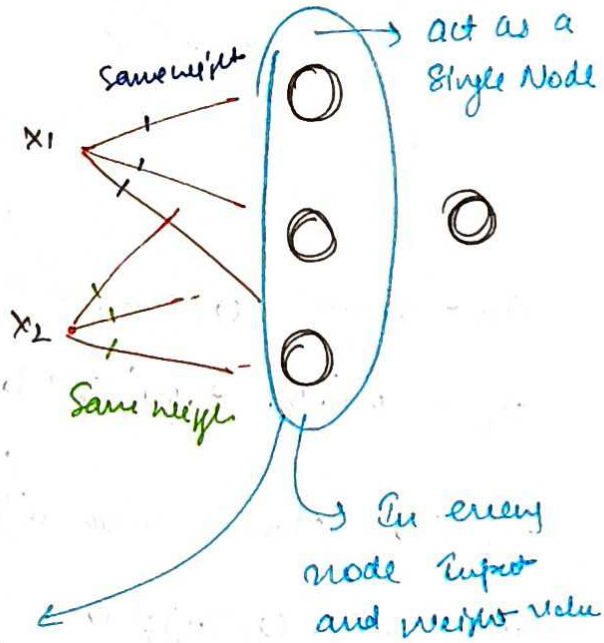
$$\frac{\partial L}{\partial w'_{21}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_{11}} \frac{\partial a_{11}}{\partial z_{11}} x_2$$

$$\frac{\partial L}{\partial w'_{22}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_{12}} \frac{\partial a_{12}}{\partial z_{12}} x_2$$

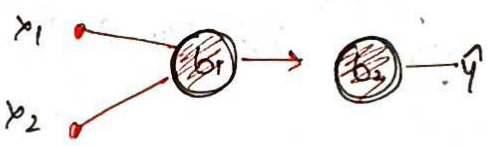
↑ Same ↑



Both weight are same. So, it act as a single weight

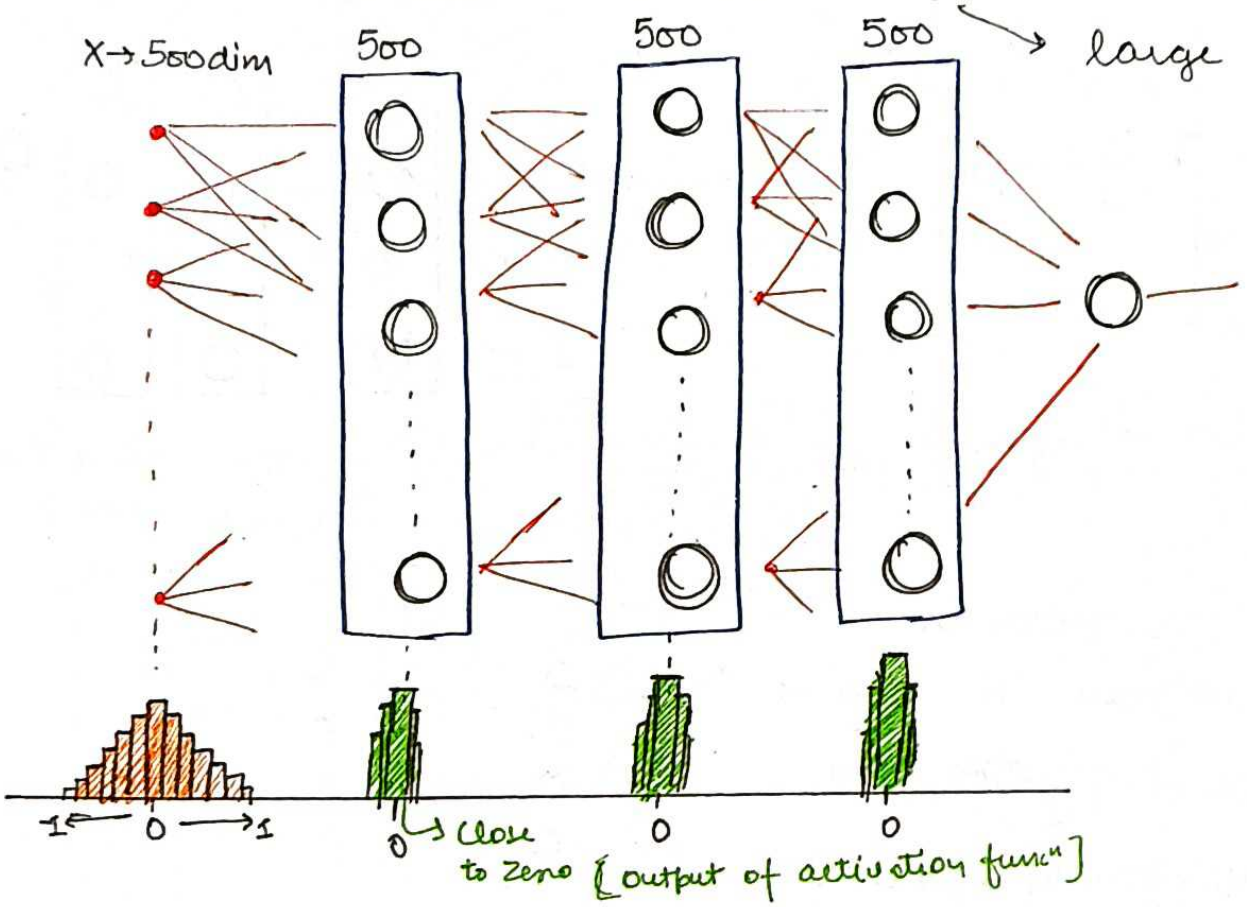


This model convert into linear model

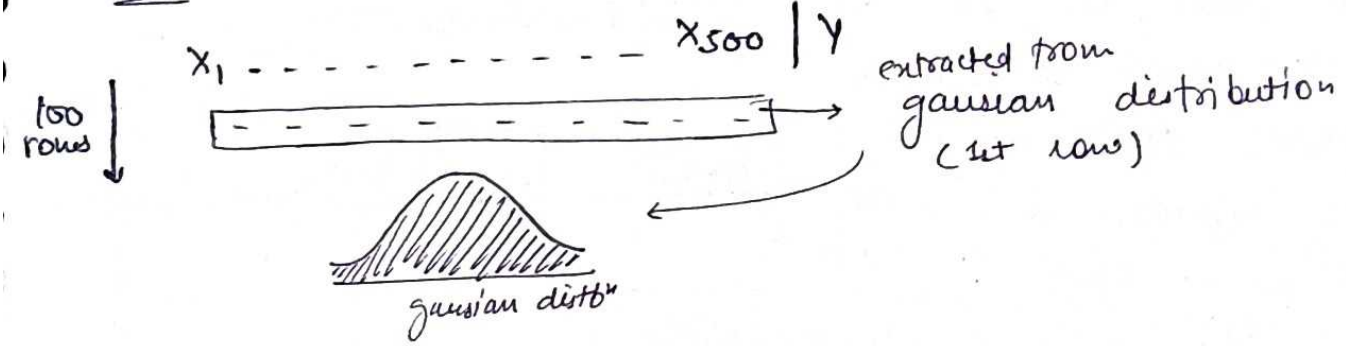


* So, Don't initialize weight and bias with 0 value and any constant value. Do initialize with random.

Case 3 - Random Initialization → small / large



Data ↓



Weight Initialize = random

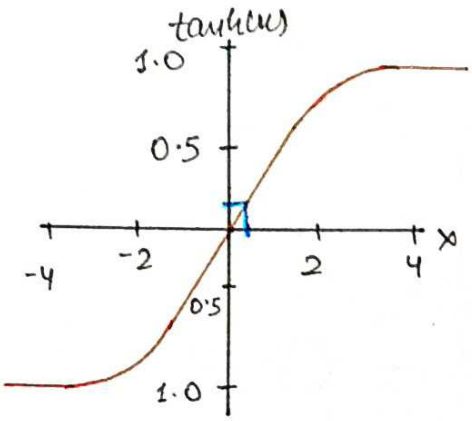
np.random.randn (500, 500) * 0.01 → after multiplying with 0.01 → weight small

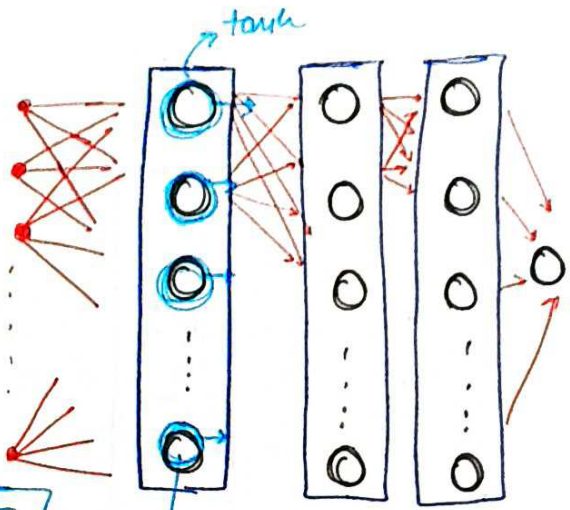
bias = 0

x_i lies bet" → -1 to 1

when we do $\sum w_i x_i$ → output is also small number.

(z) → is small



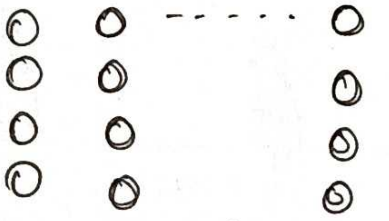


$x_i \rightarrow [1, -1]$

Weight also small

Output of activation function near to or close to zero.

When increase the layers during backpropagation time, Small number will multiple eg:- $(0.1 \times 0.2 \times 0.1 \ldots)$ then gradient will be zero or close to zero. So, when we update the weight And No update in weight When is equal to Wold



layer increase

backpropagation

$$W_{new} = W_{old} - \eta \frac{\partial L}{\partial W}$$ → Close to zero

* **Again Vanishing Gradient problem**

Summary

1. Initializing small weights → 0.0007
   And layer is deep (large) or not much large
   still facing Vanishing gradient problem in

   tanh → High Vanishing Gradient probln

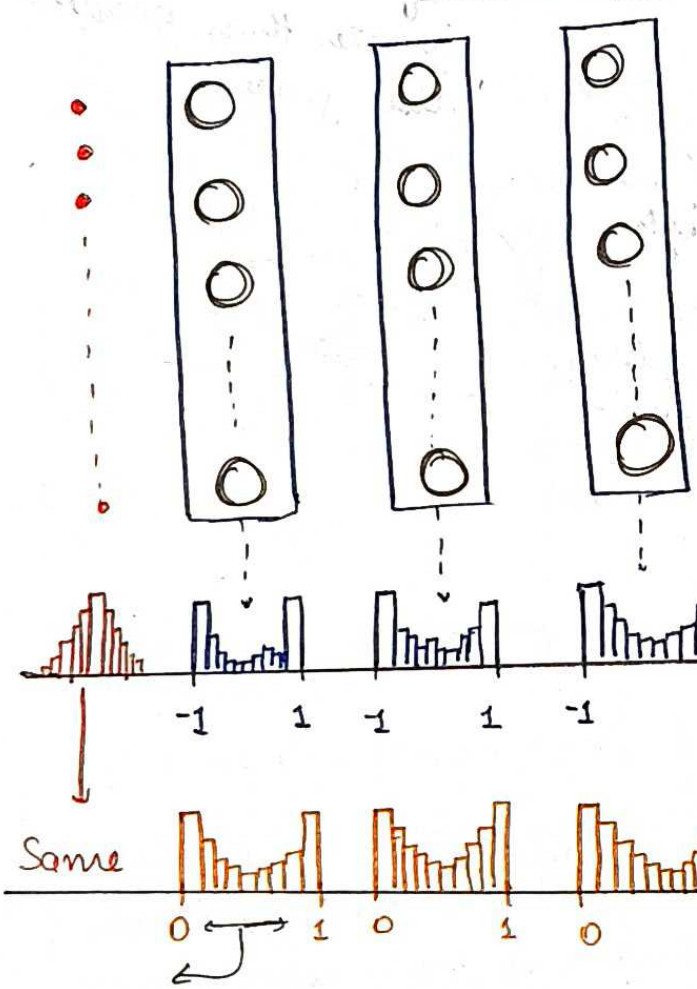
   Sigmoid → less than tanh but still face same problem.

In Relu → small value → take same vanishing gradient problem (not strong) but less than tanh and sigmoid

but training will be very slow and convergence also very slow. (large epochs need)

cz values are not zero.

Random Initialization (large value)



np.random.randn(500, 500)

→ [0-1]

this range is big for deep learning

tanh / sigmoid

Saturation → means

let $\sum w_i x_i \to 100$

↓

Start saturating

if value is high so take high or negative very low, so take negative very low value

In histogram → Maximum value in lower (0) and higher (1) in tanh and Relu.

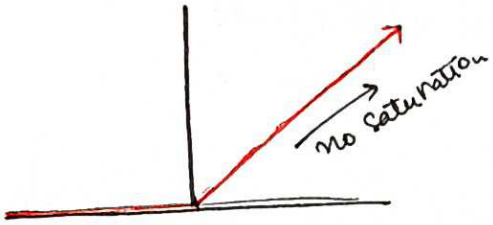
This cause → slow training

→ Vanishing gradient problem (in worst case)

Saturation → If weight is large (high positive for every node) $(\sum w_i x_i) \to 100$ send to activation function $(g(z))$. So, output is high [tanh → 1, Sigmoid → 1]

If weight is highest negative value → after activation function value is also negative (tanh = -1, Sigmoid → 0)

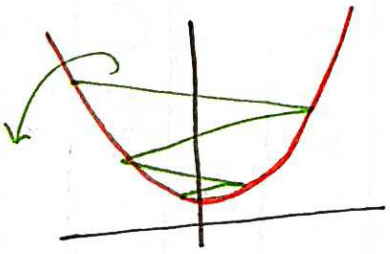
Relu → Relu is non staturating function in +x.



if $\sum W_i x_i$ is 250
then Relu output is 250
↓
If this number is greater then gradient also grater.

Update → $W = W - \eta \frac{\partial L}{\partial W}$

gradient is high then changes are also high



Not Initialize with ↓

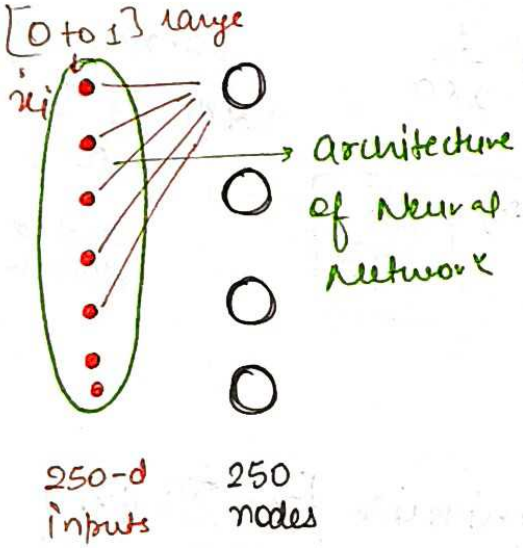
(i) Zero
(ii) non-zero
(iii) Small random
(iv) Large random

What can be done

→ Heuristics (Jugad) → practical solution

- tanh
  - Xavier/Glorot → Normal
  - Xavier/Glorot → Uniform
- He Init → Normal
- He Init → Uniform
- Relu

Intuition



Small weight (-0.01) to (0.01) ← constant

`np.random.randn(250, 250) * 0.01`

if we multiply x_i with w_i and add all $(x_i w_i)$ which is 250. So, the output is very small. And send into activation and activation convert into 0.

Large Weight (-3 to 3)

`np.random.randn(250, 250) * 1` ← constant

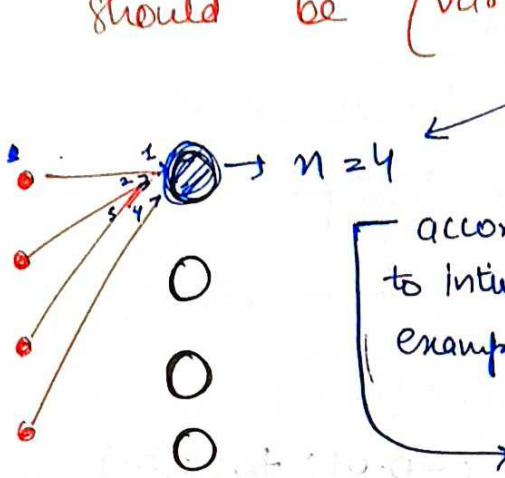
multiply weight with x_i $(w_i x_i)$ and all the 250 multiply values and add lies betn (-250 to 250). We use send these values into activation function. Activation function start saturating:

→ If value is -250 and this value reach negative max and 250 value reach positive max.

Why these problem are generating?

because of $z = \sum x_i w_i$ formula and we can not multiply with constant. Multiple the Number → it should be dependent on architecture of NN.

So, Scientist decided that which number [np.random.randn(250, 250)] generate in this code it should be [variance = 1] generated number's variance



↳ this bhi layer ka weight initialize kar rahe ho uske pichle layer mai kitne nodes hai. or uss particular node ko kitne input mil rahe.

according to intuition example

$\text{variance} = \frac{1}{n} = \frac{1}{250}$

So, multiply with $\sqrt{\frac{1}{250}}$

$np.random.rand(250, 250) * \sqrt{\frac{1}{250}}$ → Standard deviation

Intuition of variance $= \frac{1}{n}$

if $n \rightarrow \uparrow\uparrow$ then I want to minimize $z = \sum w_i x_i$

↳ no. of inputs

So, if n is greater → $\frac{1}{n}$ (multiply with weight) and individual weight is small.

another formula

$\sqrt{\frac{2}{\text{fan-in} + \text{fan-out}}}$

fan-in: No. of coming input

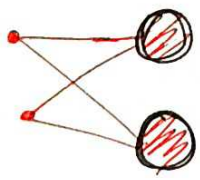
fan-out: No. of outgoing output

If n is smaller then individual weight is higher.

Let only 2 input

Xavier init (Normal) → $\sqrt{\frac{1}{\text{fan-in}}}$ → most of the time this formula use

fan-in: No. of inputs coming to the node.



→ $np.random.randn(2,2) * \sqrt{\frac{1}{2}}$

Uniform Distribution

Xavier Uniform

$[-limit, limit]$ $\quad limit = \sqrt{\frac{6}{fan\text{-}in + fan\text{-}out}}$

He Uniform

$[-limit, limit]$ $\quad limit = \sqrt{\frac{6}{fan\text{-}in}}$ $\quad$ [code in weight initialization]