Vector :- Magnitude + Direction

Unit Vector

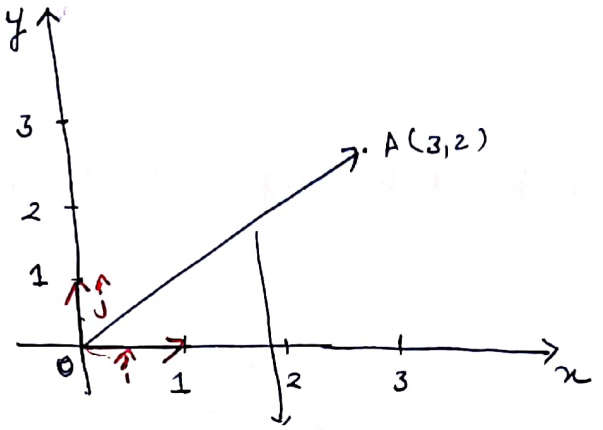
Magnitude + Direction

↓

value of magnitude is 1

aka directional vectors.

as they provide direction to a particular magnitude.



Unit vector in the direction of x is defined as $\hat{i}$

If I want to represent a vector in $\hat{i}$ direction

$\rightarrow \hat{i} = 3\hat{i}$

direction to this line

$\vec{OA} \rightarrow 3\hat{i} + 2\hat{j} = \vec{OA}$ → Position Vector

$$\frac{3}{\sqrt{13}}\hat{i} + \frac{2}{\sqrt{13}}\hat{j} = \hat{OA}$$ Unit vector.

magnitude of $\vec{OA} = |\vec{OA}| = \sqrt{3^2 + 2^2} = \sqrt{13}$

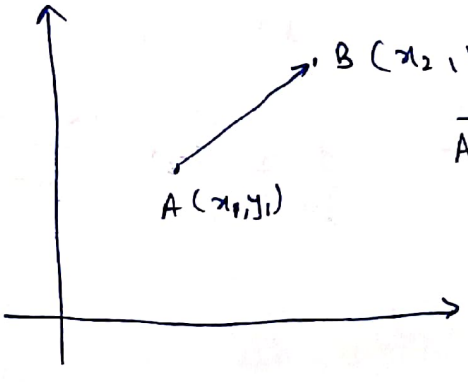
Lets say the unit vector in direction of $\vec{OA}$ is $\hat{a}$.

$$\vec{OA} = |\vec{OA}|\hat{a}$$

$$\frac{\vec{OA}}{|\vec{OA}|} = \hat{a}$$

$$\vec{v} = x\hat{i} + y\hat{j} = \begin{bmatrix} x \\ y \end{bmatrix}$$

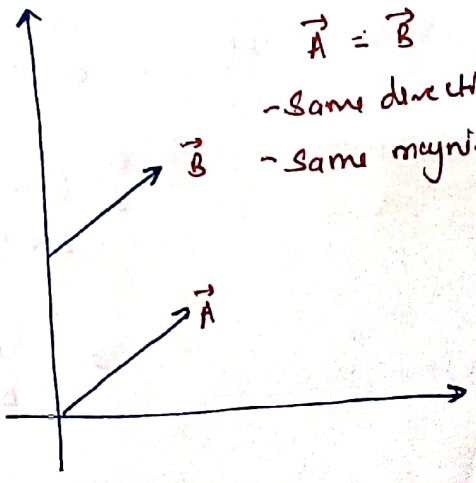
$$\vec{v} = x\hat{i} + y\hat{j} + z\hat{k} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$



$$\vec{AB} = (x_2 - x_1)\hat{i} + (y_2 - y_1)\hat{j}$$

+ Condition for two vectors to become equal.

When magnitude and direction of two vectors are same then it implies those two vectors are equal.



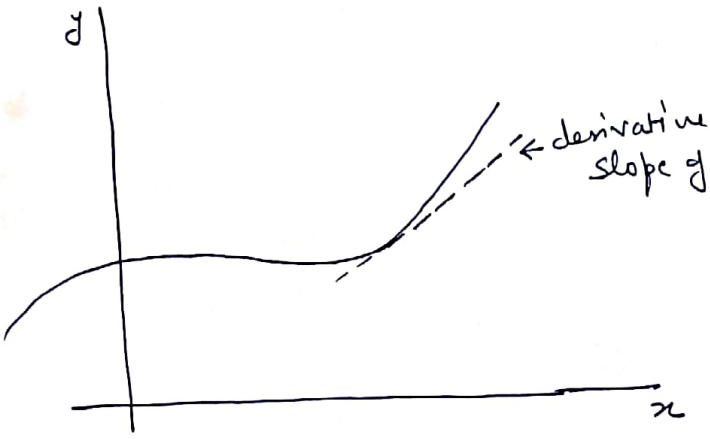
$\vec{A} = \vec{B}$

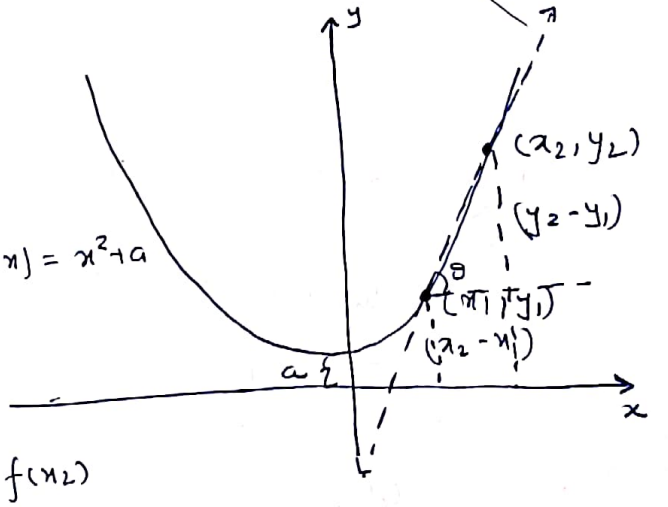
- Same direction
- Same magnitude

$y = f(x)$ — x: Independent variable; y: dependent variable

$\frac{dy}{dx} = \frac{\text{change in } y}{\text{small change in } x}$

Graphically :- slope of the tangent to the graph at a given point.





slope = $\frac{y_2 - y_1}{x_2 - x_1}$

slope = $\frac{\text{change in } y}{\text{small change in } x}$

$y_2 = f(x_2)$

$y_1 = f(x_1)$

slope = $\tan\theta = \frac{\Delta y}{\Delta x}$

$= \frac{x_2^2 - x_1^2}{x_2 - x_1} = x_1 + x_2$

$\boxed{\text{slope} = \lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x} = \frac{dy}{dx}}$

by definition of differentiation

$\frac{dy}{dx} = \lim_{\Delta x \to 0} \frac{y_2 - y_1}{\Delta x}$

$= \lim_{\Delta x \to 0} \frac{f(x_2) - f(x_1)}{\Delta x}$

$= \lim_{\Delta x \to 0} \frac{f(x_1 + \Delta x) - f(x_1)}{\Delta x}$

$\frac{d}{dx} x^n = n x^{n-1}$

$\frac{d}{dx} e^x = e^x$

$\frac{d}{dx} \frac{u}{v} = \frac{u'v - v'u}{v^2}$

Partial Differentiation

~~$y = fc$~~

$z = f(x,y) = x^2 + y^2$

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

$\frac{\partial f}{\partial x} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x, y) - f(x,y)}{\Delta x}$

$\frac{\partial f}{\partial y} = \lim_{\Delta y \to 0} \frac{f(x, y+\Delta y) - f(x,y)}{\Delta y}$

Use of Partial Differentiation in Deep Learning

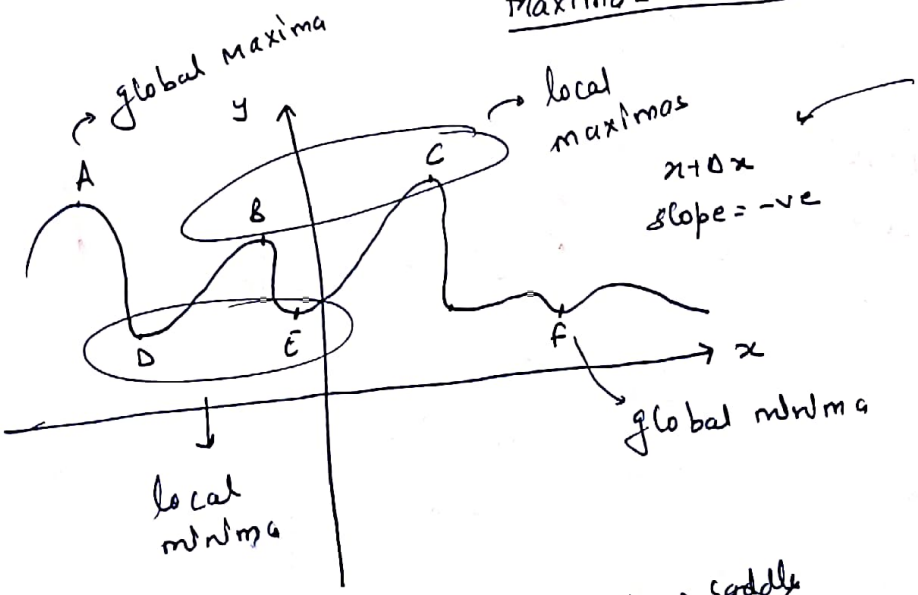
loss fn = f(weights, biases)

↳ our objective is to minimize this function

Loss fn = $f(w_1, w_2, \dots w_n, b_1, b_2, \dots b_n)$

* Partial derivate with one weight or bias and find out the change

Maxima & Minima



A, B, C = points of maxima

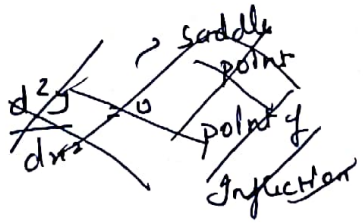
D, E, f = points of minima

↓

x+Δx slope = +ve

① $\frac{dy}{dx}$ at maxima & minima $= 0$

② $\frac{d^2y}{dx^2}$ → double derivate



at $x = c_1$

$\frac{d^2y}{dx^2} > 0 \Rightarrow$ point of minima at c_1

at $x = c_2$

$\frac{d^2y}{dx^2} < 0 \Rightarrow$ point of maxima at $x = c_2$

$f(x) = x^2$

Critical point $x = 0$

$\frac{dy}{dx} = 2x = 0$

$\frac{d^2y}{dx^2} = 2 > 0$ minima

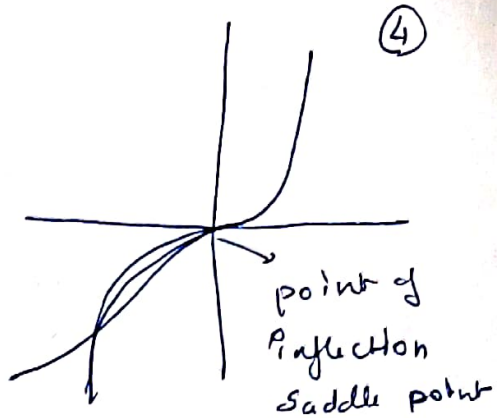
$f(x) = x^3$

~~at x =~~

$\frac{dy}{dx} = 3x^2$

at $x = 0$

$\frac{dy}{dx} = 0$



$\frac{d^2y}{dx^2} = 6x \neq 0$ (at $x = 0$)

Q Where used in Deep Learning?

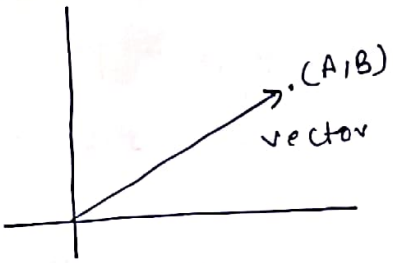
Reduction in loss function → lowest value of the error → global minima

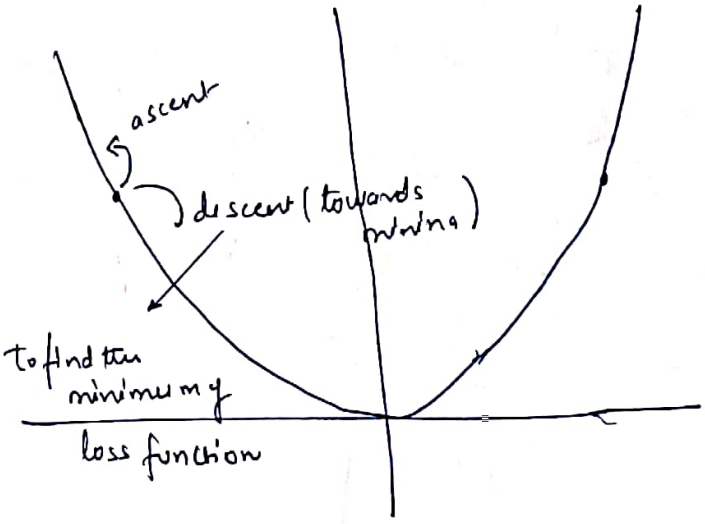
Gradient Descent

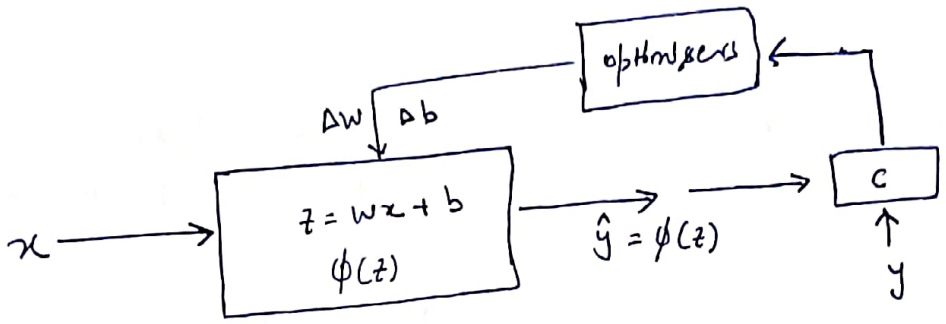
$z = f(x,y)$

$\nabla f(x,y) = \frac{\partial f}{\partial x}\Big|_y \hat{i} + \frac{\partial f}{\partial y}\Big|_x \hat{j}$

$= A\hat{i} + B\hat{j}$







$w = w + \Delta w$

$b = b + \Delta b$

$C = (y - \phi(z))^2$

$= (y - \phi(wx+b))^2$

$C \rightarrow f(w,b)$ ∵ C is nothing but function of weights and biases

optimisers — Gradient descent
- SGD
- Adams

$C = \cancel{y+t}\,(y - \phi(wx+b))^2$

Assumptions (i) treat y as a constant $K = 1$

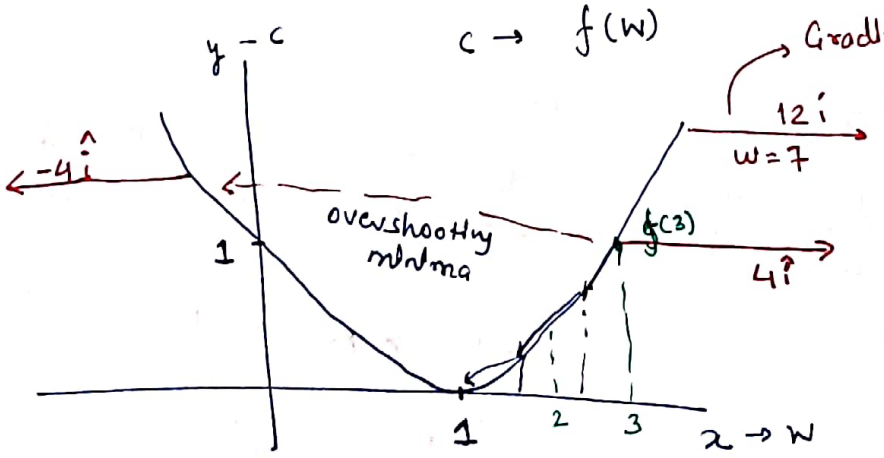
(ii) bias (b) ≈ 0

(iii) $\phi(wx+b) = \phi(wx) \simeq w$; ϕ → sigmoid fⁿ 0-1, RELU 0-∞

w → some float value

∵ $C = (1-w)^2$

$C \rightarrow f(w)$



Gradient ascent

12 î

w = 7

Step 1 :- Randomly initialize weights

let's say w = 3

C(3) = 4

f(3)

4î

$\nabla C(w=3) = \frac{\partial C}{\partial w}\,\hat{i}$

$= -2(1-w)\,\hat{i}$

$= 2(w-1)\,\hat{i}$

$\nabla C(w=3) = 4\,\hat{i}$

Assumption

(i) Let $\Delta w = 4$ $\quad w = w + \Delta w$ $\quad w = 7$] leads to gradient ascent

(ii) $w = w - \Delta w$ $\quad w = -1$ $\quad \nabla c\,(w = -1) = -4\hat{i}$] overshooting minima

∴ we need to use learning rate which is nothing but small number we multiply with gradients. We take the fraction of gradients.

$$\Delta w = -\eta \nabla c \qquad \eta = 0.1 \quad (0 \leftrightarrow 1)$$

$$w = w - \eta \nabla c = 3 - 0.1 \times 4 = 2.6$$

and every epoch we update the gradient and move towards the minima. This is called gradient descent.

Chain Rule

$$f(x) = \frac{1}{1+e^{-x}} \qquad \frac{df}{dx}$$

Let $p = -x$ $\quad q = 1 + e^{-x} = 1 + e^{p}$

$$\frac{df}{dq} \times \frac{dq}{dp} \times \frac{dp}{dx} = \frac{d}{dq}\left(\frac{1}{q}\right) \times \frac{d}{dp}(1+e^{p}) \times \frac{d}{dx}(-x)$$

$$= -\frac{1}{q^2} \times e^{p} \times -1$$

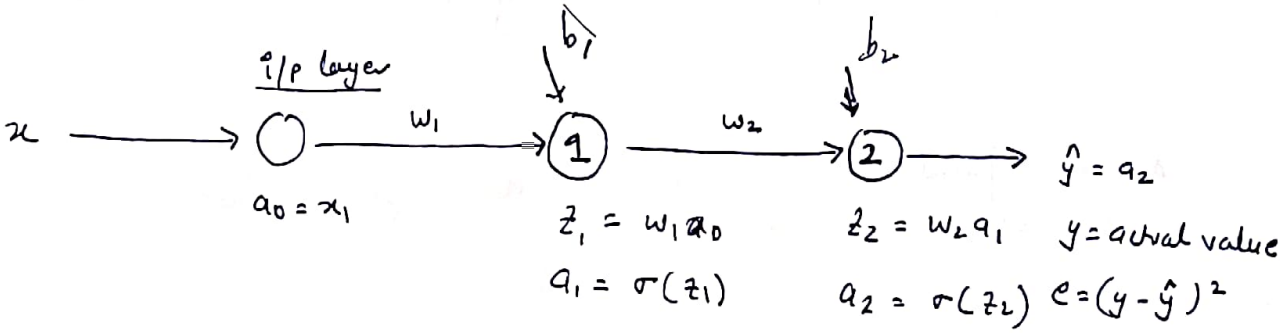
$$= \frac{e^{p}}{q^2} = \frac{e^{-x}}{(1+e^{-x})^2}$$

Backpropagation

Assumption

1) Each layer has one neuron

2) bias = 0



Weight update rule

$$w = w - \eta \nabla c = w - \eta \frac{\partial e}{\partial w}$$

Observation $e = (y - \hat{y})^2 = (y - a_2)^2$ $\quad a_2 = \sigma(z_2)$

$e = f(a_2)$ $\quad a_2 = f(z_2)$

$z_2 = w_2 a_1$

$f(w_2, a_1)$

~~Let's assume~~

$$\frac{\partial e}{\partial w_2} = \frac{\partial f(f(f(w_2, a_1)))}{\partial w_2}$$

$$\frac{\partial e}{\partial w_2} = \frac{\partial e}{\partial a_2} \cdot \frac{\partial a_2}{\partial z_2} \cdot \frac{\partial z_2}{\partial w_2}\Big|_{a_1}$$

$$= -2(y - a_2) \times \sigma(z_2)[1 - \sigma(z_2)] \times a_1$$

$$w_2 = w_2 - \eta \nabla e = w_2 - \eta \frac{\partial e}{\partial w_2}$$ ~~$= w_2 t$~~

$a_1 = \sigma(z_1)$ $\quad$ $z_1 = w_1 a_0$

$f(z_1)$ $\quad$ $f(w_1, a_0)$

$$\frac{\partial e}{\partial w_1} = \frac{\partial e}{\partial a_2} \times \frac{\partial a_2}{\partial z_2} \times \frac{\partial z_2}{\partial a_1}\bigg|_{w_2} \times \frac{\partial a_1}{\partial z_1} \times \frac{\partial z_1}{\partial w_1}\bigg|_{a_0}$$

?

At some point $\frac{\partial e}{\partial w_1} \neq 0$; no weight update

Assumption update, bias $\neq 0$

$$\frac{\partial e}{\partial w_2} = \frac{\partial e}{\partial a_2} \times \frac{\partial a_2}{\partial z_2} \times \frac{\partial z_2}{\partial w_2}$$

$$\frac{\partial e}{\partial b_2} = \frac{\partial e}{\partial a_2} \times \frac{\partial a_2}{\partial z_2} \times \frac{\partial z_2}{\partial b_2}$$

$z_2 = w_2 a_1 + b_2$

$\frac{\partial z_2}{\partial w_2} = a_1$ $\qquad$ $\frac{\partial z_2}{\partial b_2} = 1$

* General Problems in training Neural Network

1) Vanishing and Exploding Gradient
2) Lot of data required for training
3) Slow training issues
4) Risk of overfitting

$$w_{new} = w_{old} - \eta \nabla C$$

↳ lets say the value of this is decreasing and is very minute, weight update will become non significant

$\Delta w \approx 0$] → vanishing gradient issue

If this gradient becomes very huge in this case

$\Delta w \uparrow\uparrow$ (very high) →

we may divert from the soln

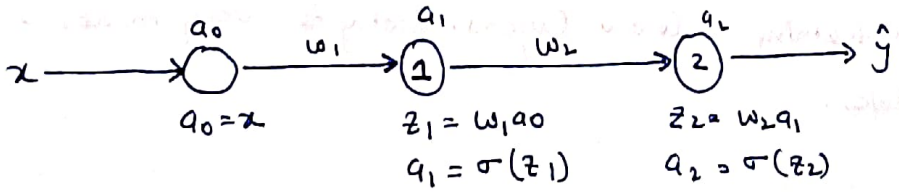
Exploding gradient.

* Neural network will overfit in case of small datasets. In order to avoid this we tend to provide a huge dataset.

Slow training Issues

- Increase the size of Neural Networks
  - ↳ Increase in number of hidden layers
  - ↳ Increases the size of the model.

Backpropogation Equation Brief



$$\frac{\partial e}{\partial w_2} = \frac{\partial e}{\partial a_2} * \frac{\partial a_2}{\partial z_2} * \frac{\partial z_2}{\partial w_2} \quad \text{—①}$$

$$\frac{\partial e}{\partial w_1} = \frac{\partial e}{\partial a_2} * \frac{\partial a_2}{\partial z_2} * \frac{\partial z_2}{\partial a_1} * \frac{\partial a_1}{\partial z_1} * \frac{\partial z_1}{\partial w_1} \quad \text{—②}$$

Observations: eq① and eq② is nothing but product of Ratios. There ratios can become greater than 1 or they can be between 0 & 1.

1st case $0 < \text{Ratio} < 1$

eq①

$$\frac{\partial e}{\partial w_2} = \underbrace{\frac{\partial e}{\partial a_2}}_{0.1} * \underbrace{\frac{\partial a_2}{\partial z_2}}_{0.2} * \underbrace{\frac{\partial z_2}{\partial w_2}}_{0.3} \approx 0.006$$

$$w_2 = w_1 - \eta\, 0.006 \qquad \text{Lets } \eta = 0.1$$

$$w_2 = w_1 - 0.1(0.006)$$

$$\boxed{w_2 = w_1 - 0.0006}$$ — almost negligible change

$\Rightarrow$ There will be no weight update significantly

* This is known as vanishing gradient Problem that means very minute or negligible weight update.

Assumption :- All the ratios in $\frac{\partial e}{\partial w_2}$ equation② are b/w 0 and 1.

$$\frac{\partial e}{\partial w_2} \approx \text{negligible}$$

This also leads to slow training of lower layers. Owing to very minor & negligible weight updates.

Observation 2nd

If all the ratio are greater >> 1

eq①

$$\frac{\partial e}{\partial w_2} = \frac{\partial e}{\partial a_2} \times \frac{\partial a_2}{\partial z_2} \times \frac{\partial z_2}{\partial w_2} \approx 6000$$

$$\frac{\partial e}{\partial a_2} \approx 10 \qquad \frac{\partial a_2}{\partial z_2} \approx 20 \qquad \frac{\partial z_2}{\partial w_2} \approx 30$$

$w_2 = w_2 - \eta\, 6000$ very large weight update

Similarly it will be further greater in case of eq②

solution will diverge

This problem was first observed in 2010 by Xavier Glorot & Yoshua Bengio in their paper "Understanding the difficulty of training deep feed forward neural network"

Why vanishing and exploding gradient problem occur?

1) choice of activation function $\frac{\partial a_2}{\partial z_2}$] derivative of activation function

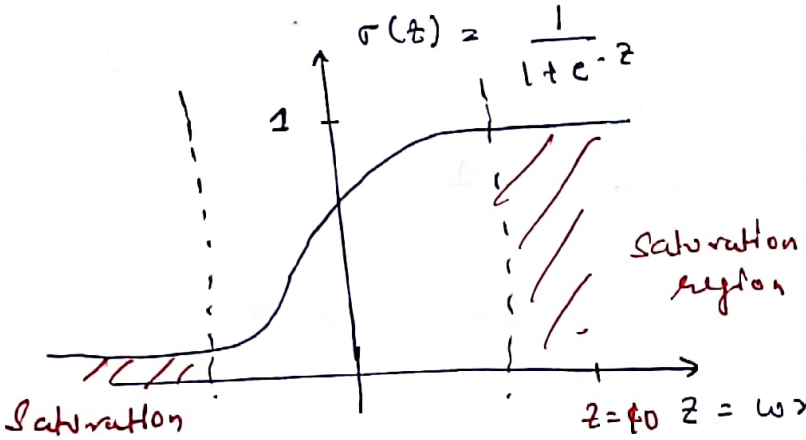
2) Weight initialization technique $\frac{\partial z_2}{\partial w_2}$ |

Activation function

It brings non linearity and help us to converge to the solution i.e help us find out the minima of loss function.

$$\frac{\partial e}{\partial w} = \frac{\partial e}{\partial a} \times \left(\frac{\partial e}{\partial z}\right) \times \frac{\partial z}{\partial w}$$

↳ this will bring convergence



(Let's assume bias is not there)

Case 1:-

Initialised weight

$w = 10$, assume $x = 1$

$z = wx = w \cdot 1 = 10$

$e \simeq 2.73$

$\sigma(z=10) = \frac{1}{1+e^{-10}}$ $\quad e^{-10} \simeq 0$

$\simeq \frac{1}{1+0} = 1$

at point $z = 10$ $\quad \frac{\partial \sigma(z)}{\partial z} = \sigma(z)\{1-\sigma(z)\}$

$\frac{\partial \sigma}{\partial z}(z=10) = 1 \times (1-1) = 0$

$\frac{\partial e}{\partial w} = \frac{\partial e}{\partial a} \cdot \cancel{\frac{\partial a}{\partial z}}^{\,0} \cdot \frac{\partial z}{\partial w} \simeq 0$

No weight update / negligible weight update

Case 2 :

$z = -10$ $\quad w = -10$ $\quad x = 1$

$\sigma(z=-10) = \frac{1}{1+e^{10}} \simeq 0$

$\frac{\partial \sigma(z)}{\partial z} = \sigma(z)\{1-\sigma(z)\} = 0$

$\frac{\partial e}{\partial w} = \frac{\partial e}{\partial a} \times \cancel{\frac{\partial a}{\partial z}}^{\,0} \times \frac{\partial z}{\partial w}$

i.e if we initialize weight very high or very low in case of sigmoid function results in vanishing gradient.

Case 3 $z = 0$ / $w = 0$ $x = 1$

$$\sigma(z=0) = \frac{1}{1+e^0} = 0.5$$

$$\frac{\partial \sigma(z)}{\partial z} = \sigma(z)\{1-\sigma(z)\} = 0.5 \times 0.5 = 0.25$$

$$\frac{\partial e}{\partial w} = \frac{\partial e}{\partial a} \times \frac{\partial a}{\partial z} \times \frac{\partial z}{\partial w}$$

$$= 0.25 \left[\frac{\partial e}{\partial a} \times \frac{\partial z}{\partial w}\right]$$

i.), there will be significant weight update.

⇒ Through the above example we understood that both activation function and weight initialisation contribute to vanishing and exploding gradient

⇒ Previously we use to randomly initialise the weights.

$$\frac{\partial}{\partial x}\frac{u}{v} = \frac{u'v + v'u}{v^2}$$

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

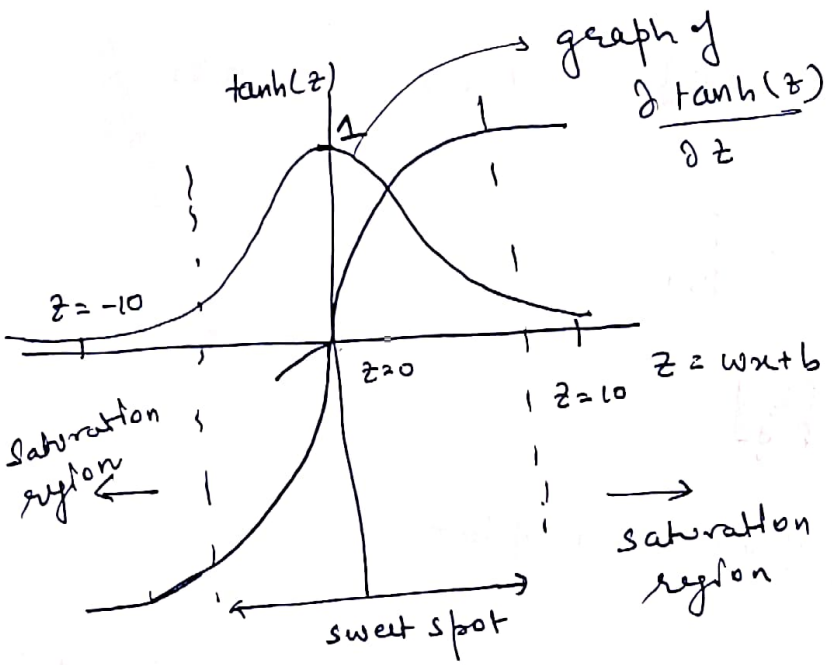
$$\frac{\partial \tanh(z)}{\partial z} = \frac{(e^z + e^{-z})(e^z + e^{-z}) - (e^z - e^{-z})(e^z - e^{-z})}{(e^z + e^{-z})^2}$$

$$= \frac{(e^z + e^{-z})^2 - (e^z - e^{-z})^2}{(e^z + e^{-z})^2} = 1 - \left(\frac{e^z - e^{-z}}{e^z + e^{-z}}\right)^2$$

$$= 1 - (\tanh(z))^2$$

$$\text{Relu}(z) = \begin{cases} 0 & z < 0 \\ z, & z \geq 0 \end{cases}$$

$$\boxed{\frac{\partial\, \text{Relu}(z)}{\partial z} = \begin{cases} 0 & z < 0 \\ 1 & z \geq 0 \end{cases}}$$



Case 1 Let $w = 10$, $b = 0$, $x = 1$

Weight Initialize

$$\tanh(z=10) = \frac{e^{10} - e^{-10}}{e^{10} + e^{-10}} \approx 1$$

$$\left.\frac{\partial \tanh(z=10)}{\partial z}\right|_{z=10} = 1 - (\tanh(z=10))^2$$

$$\approx 0$$

Hence issue of vanishing gradient

Case 2 $z = -10$

$$\tanh(z=-10) = \frac{e^{-10} - e^{10}}{e^{-10} + e^{10}} \approx -1$$

$$\left.\frac{\partial \tanh(z=-10)}{\partial z}\right|_{z=-10} = 1 - (\tanh(z=-10))^2 \approx 0$$

Case 3 $w = 0$ $b = 0$ $x = 1$

$$\tanh(z=0) = \frac{e^0 - e^0}{e^0 + e^0} = 0$$

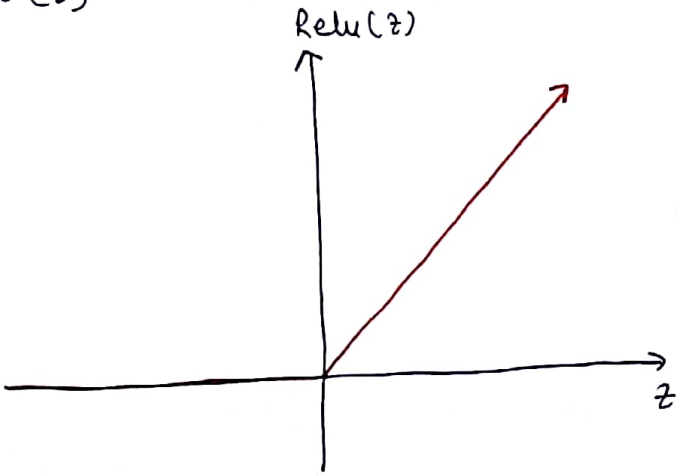
$$\left.\frac{\partial \tanh(z=0)}{\partial z}\right|_{z=0} = 1 - (0)^2 = 1$$

$$\frac{\partial e}{\partial w} = \frac{\partial e}{\partial a} \times \frac{\partial a}{\partial z}^{1} \times \frac{\partial z}{\partial w}$$

↳ weight update will be significant

In forward propogation information flows through $\tanh(z)$ in backward propogation information flows through $\frac{\partial \tanh(z)}{\partial z}$ (derivative of $\tanh(z)$)

Relu(z)



Case 1 $w = 10$ $x = 1$

$$\frac{\partial\, Relu(z=10)}{\partial z}\Big|_{z=10} = 1$$

Significant changes in weight

Case 2

$w < 0$ $x = 1$

$z < 0$

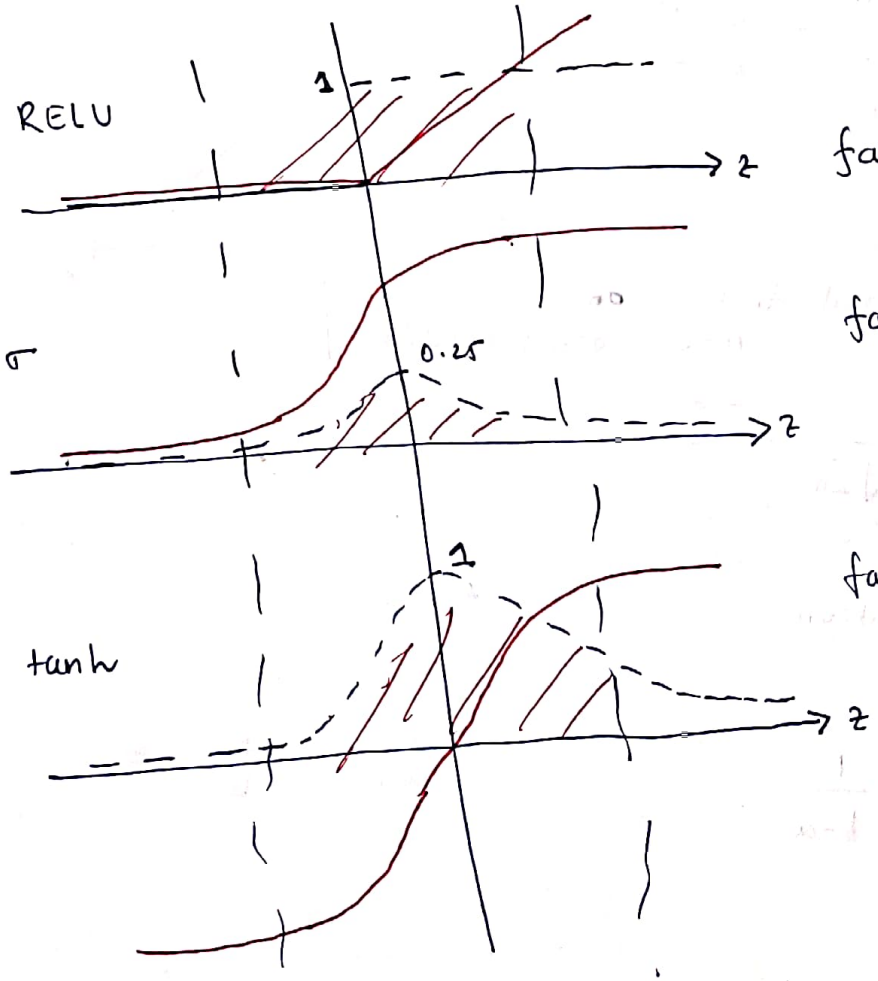
$$\frac{\partial\, Relu(z)}{\partial z}\Big|_{z<0} = 0$$

- Vanishing gradient Problem

* Observation made by the Research paper by Xavier Glorot & Yoshua Bengio

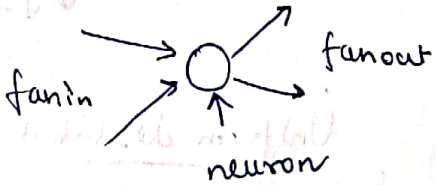
-- derivative backward

— forward.



fan in = Number of incoming edges to a layer

fan out = Number of outgoing edges from a layer



To prevent vanishing and exploding gradient:

1) fan in = fan out but not always possible to have same input & output edges.

alternative proposal :- if $\sigma_{in}^2 = \sigma_{out}^2$ (variance of weight of input edges is equal to variance of weights of output edges)

(16) $\sigma_{in}^2 = \sigma_{out}^2$: helps you maintain forward & backward information flow

Calculate

$$fan_{avg} = \frac{fan_{in} + fan_{out}}{2}$$

for sigmoid activation fn

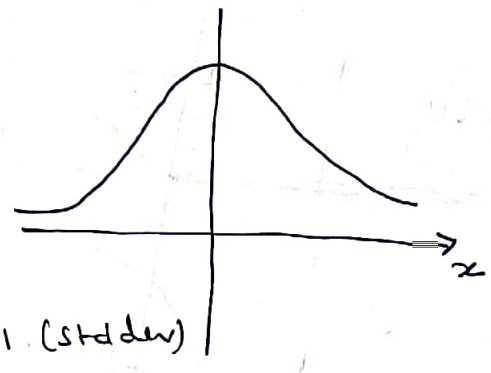
- Normal distribution with $\mu = 0$ & $\sigma = 1/fan_{avg}$
- Uniform distribution b/w $-r$ & $+r$ where $r = \sqrt{\frac{3}{fan_{avg}}}$

↓

Glorot Initialization or Xavier Initialization] default case of weight initialization in Keras.

Normal distribution

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



standard Normal distribution

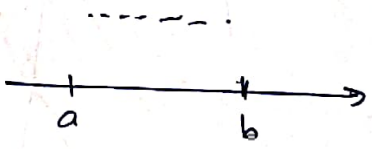
$\mu = 0$, $\sigma = 1$ (std dev)

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$$

Uniform distribution

a, b

$$f(x) = \frac{1}{b-a}$$



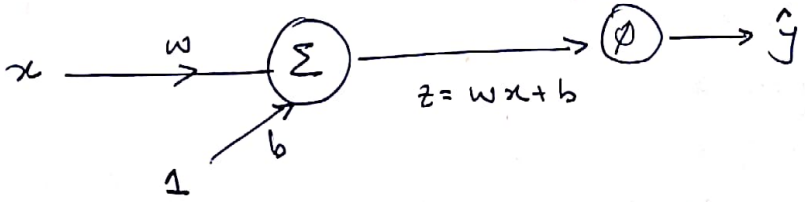
Type of Initialisation		Activation function
Xavier Glorot + Yoshua Bengio	Glorot →	None, tanh, sigmoid, softmax. σ^2 (normal) = $\frac{1}{fan_{avg}}$ variance
Kaiming He	He →	RELU and its $\sigma^2 = 2/fan_{in}$ variance normal dist
Yaan Lecan	Lecan -	SELU $\sigma^2 = 1/fan_{in}$ normal dist

if $fan_{in} = fan_{avg}$ ⇒ Glorot and Lecan initialisation are same

In tensorflow

tf.keras.dense(unit, activation function, kernal_initializer = "he_normal")

By default its glorot initialisation.



Forward Pass $\hat{y} = \phi(z)$ where $z = wx + b$

Backward Pass

$$\frac{\partial C}{\partial w} = \frac{\partial C}{\partial \phi} \times \frac{\partial \phi}{\partial z} \times \frac{\partial z}{\partial w}$$

	domain	range
sigmoid	$(-\infty, \infty)$	$(0, 1)$
	$(-\infty, \infty)$	$(0, 0.25)$

For forward pass: use $\phi(z)$ and consider its domain & range

For backward pass: use $\phi'(z)$ consider its domain and range

1. Sigmoid activation function

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

Range $\sigma(z) \in (0,1)$

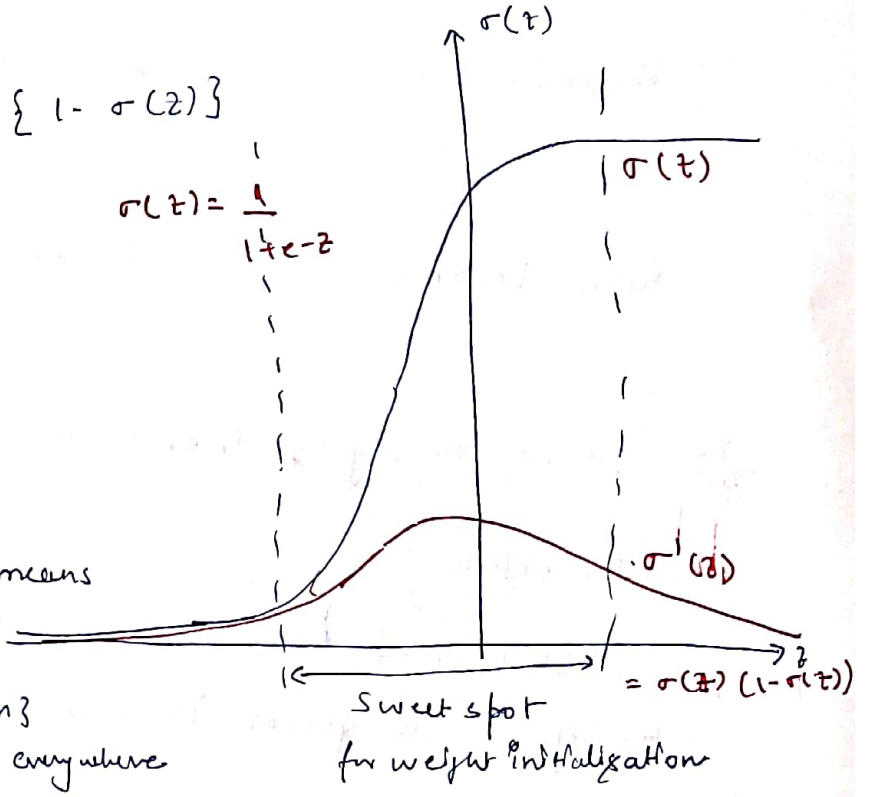
domain $z \in (-\infty, \infty)$

Backward

$$\sigma'(z) = \sigma(z)\{1-\sigma(z)\}$$

Range $\in (0, 0.25)$

domain $\in (-\infty, \infty)$



Advantages

1) It has smooth gradient that means this prevents jumps in the output value. {Continuous function} i.e it is derivable everywhere

2) Output values are range of $\sigma(z)$ b/w (0,1) this implies normalising the output of each neuron. This will help us to reduce the solution space.

Disadvantages

1) It is prone to gradient vanishing. as $\sigma'(z) \in (0, 0.25)$

2) ~~Function~~ Power operation on exponential terms makes calculations time consuming i.e it will simply increase the latency.

Mostly used in output layer

2) Hyperbolic Tangent activation function

$$\tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$ where

$\tanh(z) \in (-1,1)$ Range

Domain $(-\infty, \infty)$

$\tanh'(z) = 1 - \tanh^2(z)$

range → $\tanh'(z) \in (0,1)$

domain → $(-\infty, \infty)$

provided proper weight initialisation

* Better than sigmoid activation function as it has lesser chances of vanishing gradient
* Smooth gradient - derivable at every point
* Symmetric to origin - which leads to zero mean - it will provide equal balance to negative and positive side
* Suitable function for hidden layers.

Disadvantage

* It has saturation region where gradient becomes zero.
* It introduces latency because of exponential terms.

3) Rectified Linear Unit (RELU), most popular, highly used, most successful

Forward Pass

$ReLU(z) = \max(z, 0)$ domain $z \in (-\infty, \infty)$ Range $ReLU(z) \in [0, \infty)$

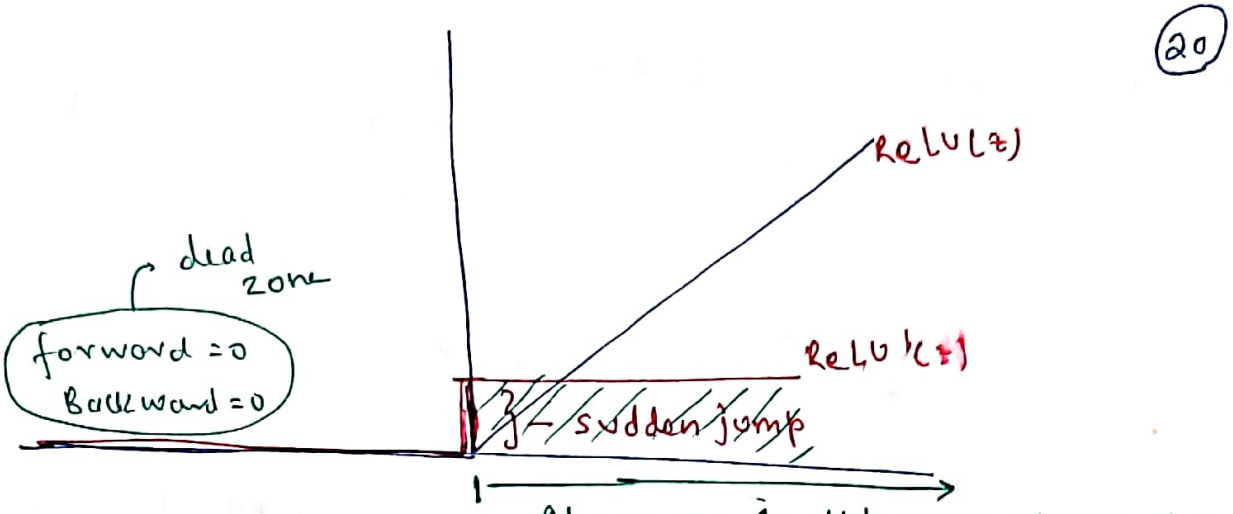
$$= \begin{cases} z & z \geq 0 \\ 0 & z < 0 \end{cases}$$

Backward Pass : Issue with derivative as this function is not continuous at 0.

$$ReLU'(z) = \begin{cases} 1 & z > 0 \\ 0 & z < 0 \end{cases}$$ → derivative is not defined at $z = 0$

Range ~~0,1~~ $\in 0,1$

domain $(-\infty, 0) \cup (0, \infty)$



Advantages

① If input is +ve ⇒ ~~the~~ There is no gradient saturation problem.

② Calculation is very fast compared to prev activation function both in forward and backward direction

Disadvantages

① For negative inputs ReLU is completely inactive ⇒ Dying ReLU issues.

② Not a zero centric function. At zero, its derivatives is not defined
↳ jump leftside

To resolve the issue of ReLU not being derivable, people have developed variants of ReLU.

④ Leaky ReLU functions. (Resolve dead zone issue)

forward

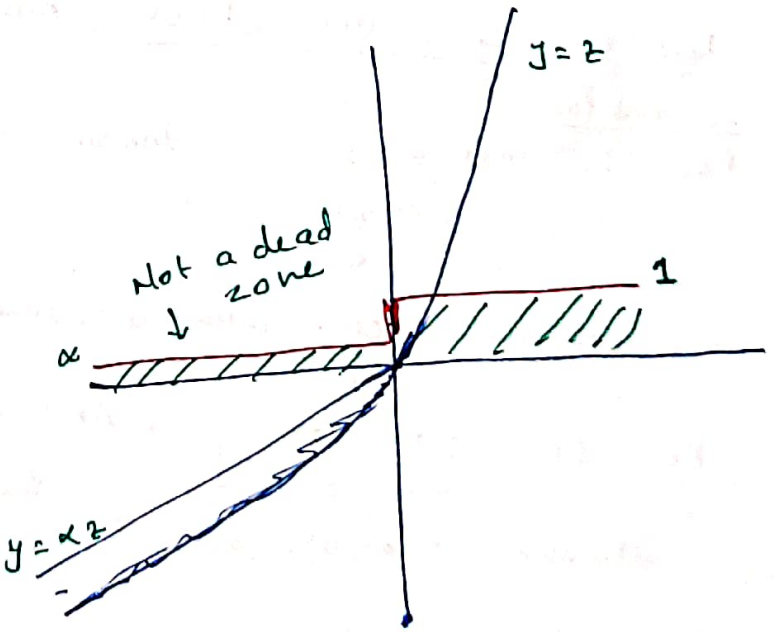
Leaky ReLU $(z) = \max(z, \alpha z)$

Leaky ReLU $(z) = \begin{cases} z & z > 0 \\ \alpha z & z < 0 \end{cases}$

domain $z \in (-\infty, \infty)$

Range $\in$ ~~$[0, \infty)$~~ $(-\infty, \infty)$

$\alpha \approx 0.01$



Backward Pass derivative

Leaky ReLU $'(z) = \begin{cases} 1 & z > 0 \\ \alpha & z < 0 \end{cases}$

not derivable at $z = 0$

domain $z \in (-\infty, \infty)$

Range $\in \alpha, 1$

(5) PReLU : Parametric ReLU

$$PReLU = \begin{cases} z & z \geq 0 \\ \alpha z & z < 0 \end{cases}$$

If $\alpha = 0 \Rightarrow$ ReLU

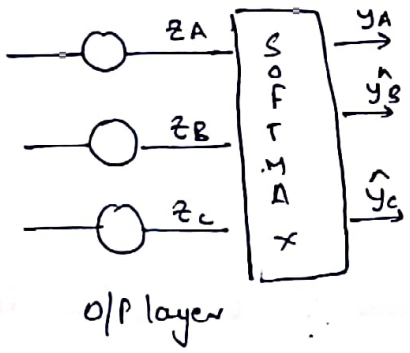
$\alpha \gg 0 \Rightarrow$ Leaky ReLU

α is learnable $\rightarrow$ PReLU

$\boxed{\alpha}$ → Learnable parameter
↳ this will also be trained.

* Disadv. Need to train extra parameter.

(6) Softmax Activation functions



O/P layer

$$\frac{e^{z_A}}{\Sigma e} \quad \frac{e^{z_B}}{\Sigma e} \quad \frac{e^{z_C}}{\Sigma e}$$

$$\hat{y}_A + \hat{y}_B + \hat{y}_C = 1$$

$$\Sigma e = e^{z_A} + e^{z_B} + e^{z_C}$$

Outcome is nothing but probability distribution

If number of classes > 2

Case study

Case 1 : binary classification {Sigmoid fn}

$$z = w_1 x_1 + w_2 x_2 + w_3 x_3 = W^T x = \theta^T x$$

Sigmoid activation fn for output layer

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

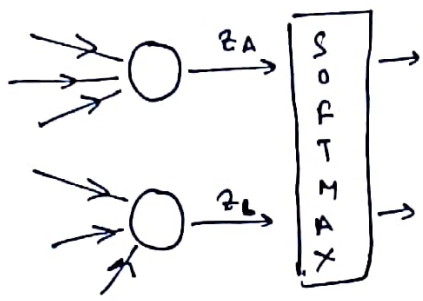
$p(y=1|x)$ ✓ probability of $y=1$ given the value of x

$$= \frac{1}{1+e^{-z}} = \frac{1}{1+e^{-\theta^T x}}$$

$$p(y=0|x) = 1 - p(y=1|x) = \frac{e^{-\theta^T x}}{1+e^{-\theta^T x}}$$

Case 2 :- Binary classification with softmax

$C = 2$



$$p(y=1|x) = \frac{e^{\theta_A^T x}}{e^{\theta_A^T x}+e^{\theta_B^T x}} = \frac{1}{1+e^{(\theta_B^T-\theta_A^T)\cdot x}} = \frac{1}{1+e^{-\theta^T x}}$$

Let $\theta_B^T - \theta_A^T = -\theta^T$

$$p(y=0|x) = \frac{e^{-\theta^T x}}{1+e^{-\theta^T x}}$$

$\Rightarrow$ These eqn are similar for sigmoid case. Hence for binary classification softmax is equivalent to sigmoid activation.

* Only drawback of softmax is that it contains exponential terms which simply implies that it computationally intensive
* Softmax is only used in output function.

$\Rightarrow$ * Remember unsolved issue of ReLU not derivable at $x=0$

(7) ELU (Exponential Linear Units)

Forward Pass

$$elu(z,\alpha) = \begin{cases} z & z \geq 0 \\ \alpha(e^z-1) & z<0 \end{cases}$$

α - scaler value and | slope of -ve section

domain $\in (-\infty, \infty)$

Range $\in (-\alpha, \infty)$

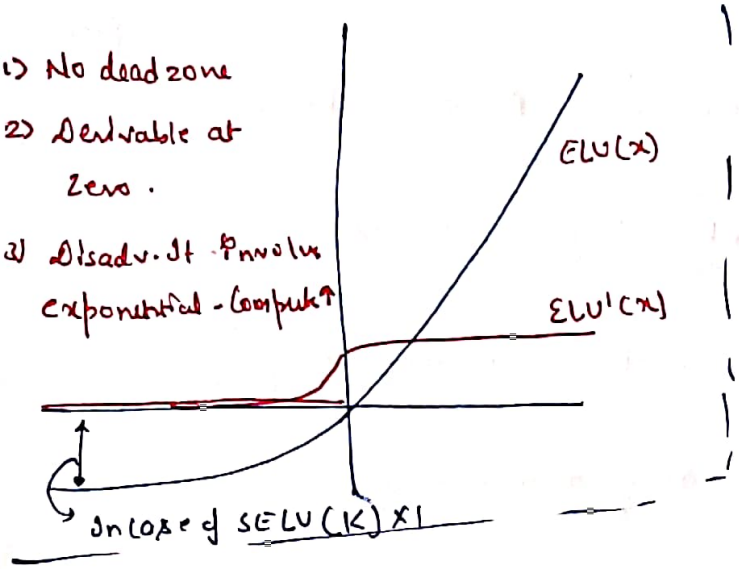
Backward Pass

$$elu(z,\alpha) = \begin{cases} 1 & z \geq 0 \\ \alpha e^z & z<0 \end{cases}$$

domain $z \in (-\infty, \infty)$

Range $\in [0,1]$

1) No dead zone
2) Derivable at zero.
3) Disadv. It involves exponential - Compute



(8) SELU - Scaled ELU

$$SELU(z,\alpha) = scale \times ELU(z,\alpha)$$
$$= K\, ELU(z,\alpha)$$

(9) SWISH

$$Swish(z) = z \cdot \sigma(\beta \cdot z) = \frac{z}{1+e^{-\beta z}}$$

β: learnable parameter

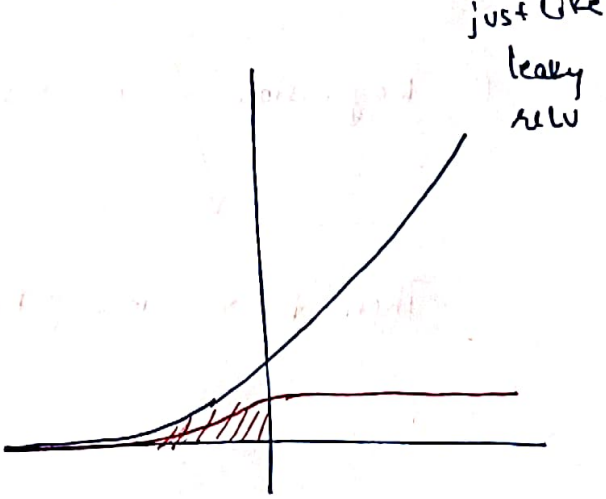
Similar to sigmoid.

domain :- $(-\infty, \infty)$

range = $(-K, \infty)$

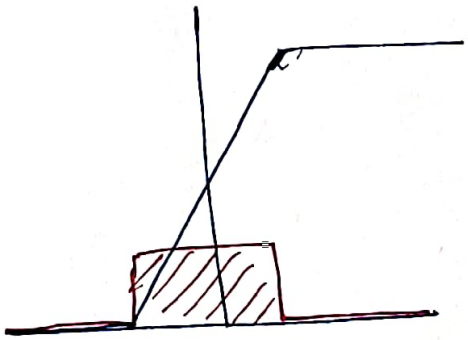
(10) SOFTPLUS

$$Softplus(z) = \log(1+e^z)$$



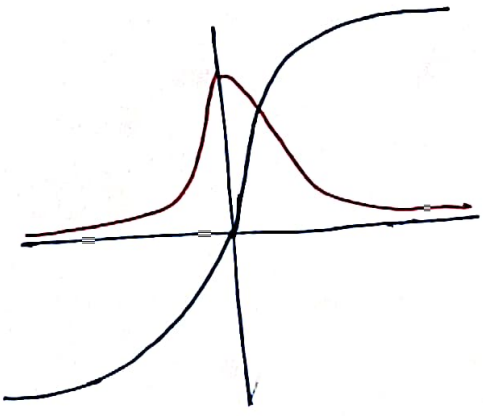
(11) Hard Sigmoid

$$\phi(z) = \begin{cases} 0 & z < -2.5 \\ 1 & z > 2.5 \\ 0.2z + 0.5 & -2.5 \le z \le 2.5 \end{cases}$$



(12) Soft sign

$$ss(z) = \frac{z}{(|z|+1)} = \begin{cases} \frac{z}{z+1} & z \ge 0 \\ \frac{z}{1-z} & z < 0 \end{cases}$$

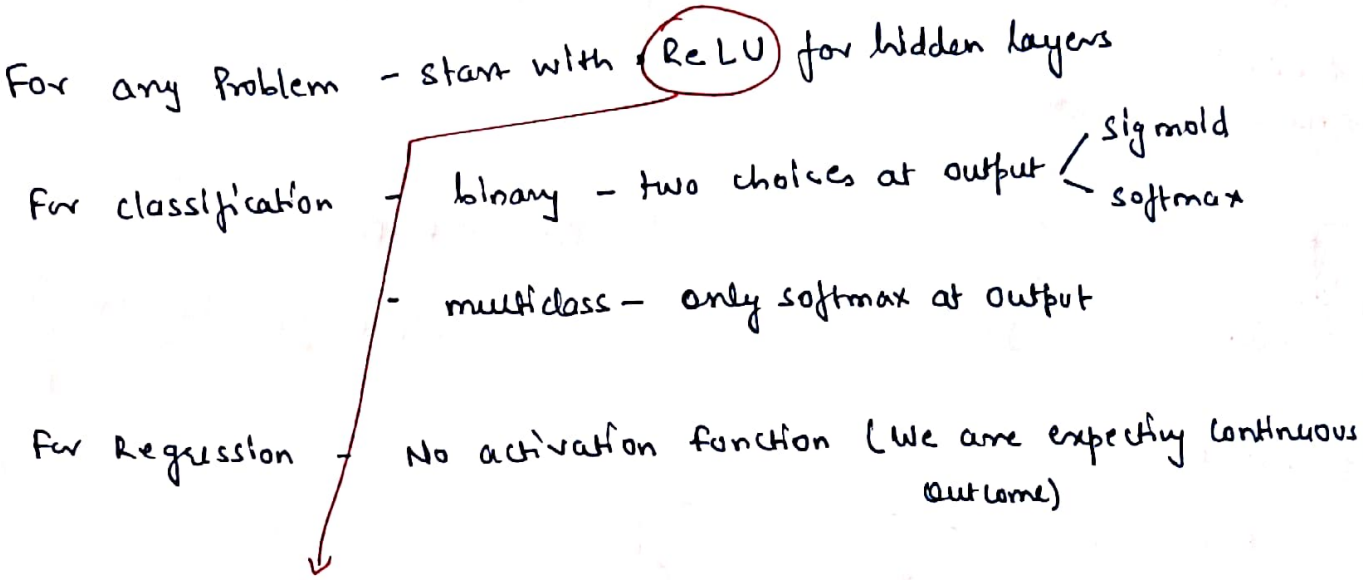


(13) Max out

$$f(x) = \max(w_1^T x + b_1, w_2^T x + b_2)$$

drawback : 1) Compute intensive as it doubles the number of parameters

Rough Idea on how to begin selection of Activation function



Then try versions of ReLU if not getting better results.