# Agenda

1) Problems in Training NN -
2) Weight initialization & activation func" basics
3) Act function.
4) Transfer learning
5) Batch Normalization

## Remaining topic

(1) Optimizers
(2) Regularization
(3) Loss function

1)

i) Vanishing & Exploding Gradients

$W = W - \eta \underbrace{\nabla C}_{\uparrow\uparrow}$

Gradients ↓↓ → $\nabla C \downarrow\downarrow$

Grad ↑↑ → $\nabla C \uparrow\uparrow$

unstable Gradients => you'll never reach to a good solution

⇓

Solution will not converge

ii) NN requires lot of data to train

2M parameters → MNIST. 2 6 6 6 1 0 —

Sol": Transfer learning or data augmentation.

iii) For complex problem statement:

Increase size of NN => increase in no of hidden layers

=> slow training

Sol": Choose a better optimizer & activation f".

iv) Risk of overfitting.

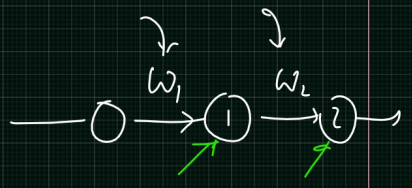
→

solution : {Regularization → Dropout}

Observation :-

$$\frac{\partial e}{\partial w_1} = \frac{\partial e}{\partial a_2} \cdot \frac{\partial a_2}{\partial z_2} \cdot \frac{\partial z_2}{\partial w_2} \quad -(1)$$

$$\frac{\partial e}{\partial w_1} = \frac{\partial e}{\partial a_2} \cdot \frac{\partial a_2}{\partial z_2} \cdot \frac{\partial z_2}{\partial a_1} \cdot \frac{\partial a_1}{\partial z_1} \cdot \frac{\partial z_1}{\partial w_1} \quad -(2)$$




$z_2 = w_2 a_1 + b$

These are product of ratios

1st case :- all the ratio $\ll 1$

eqn. (1) $\frac{\partial e}{\partial w_2} = 0.1 \times 0.2 \times 0.3 \simeq \underline{0.006}$

$$w_2 = w_2 - \eta \times 0.006$$

$$w_2 = w_2 - \underbrace{0.1 \times 0.006}_{\simeq 0}$$

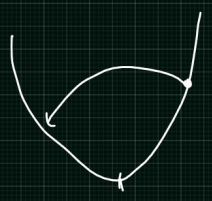
⇒ Vanishing gradient ⇒ negligible weight update

⇒ leads to train lower layer very slow

2nd case : product terms $\gg 1$

$$\frac{\partial e}{\partial w_2} = 10 \times 20 \times 30 = 6000$$

$$w_2 = w_2 - \eta\, 6000$$



⇒ Exploding gradient issue

⇒ solution will diverge

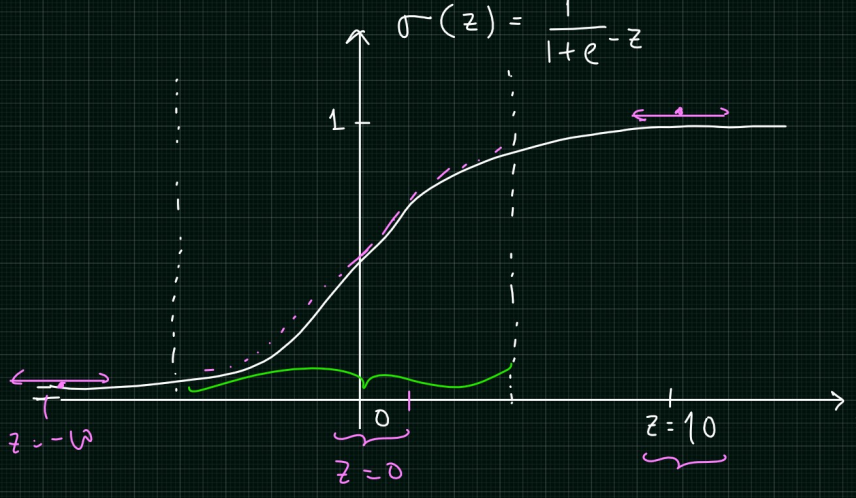
for Vanishing & Exploding gradient.

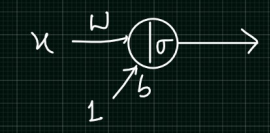
it depends on choice of activation funct.ⁿ
&
Weight initialization technique

in 2010 by
Xavier Glorot & Yoshua Bengio
in their paper
"Understanding the difficulty of training deep feed forward NN"

⇓

each layer learn at different speed.




Case 1. Initialized weight

$W = 10$ assume $\rightarrow x = 1$

$z = W \cdot x = 10 \times 1 = 10$

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

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

$e \simeq 2.73 \simeq 3$

$3^{-10} = \frac{1}{3^{10}} \simeq 0$

at $z = 10$

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

$$\sigma'(z=10) = \sigma(z=10)\{1-\sigma(z=10)\}$$

$$= 1\{1-1\}$$

$$= 0$$

$$\frac{\partial e}{\partial w_2} = \frac{\partial e}{\partial a_2} \cdot \frac{\partial a_2}{\partial z_2}^{0} \cdot \frac{\partial z_2}{\partial w_2} = 0$$

$\Rightarrow$ zero weight update

Case 2: Weight init-

$w = -10$ assume $x = 1$

$z = -10 \times 1 = -10$

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

$$\sigma'(z=-10) = \sigma(z=-10)\{1-\sigma(z=-10)\}$$

$$= 0$$

$$\frac{\partial e}{\partial w_2} = \frac{\partial e}{\partial a_2} \cdot \frac{\partial a_2}{\partial z_2}^{0} \cdot \frac{\partial z_2}{\partial w_2} \simeq 0$$

$\Rightarrow$ zero weight update

Case 3.

$w = 0$

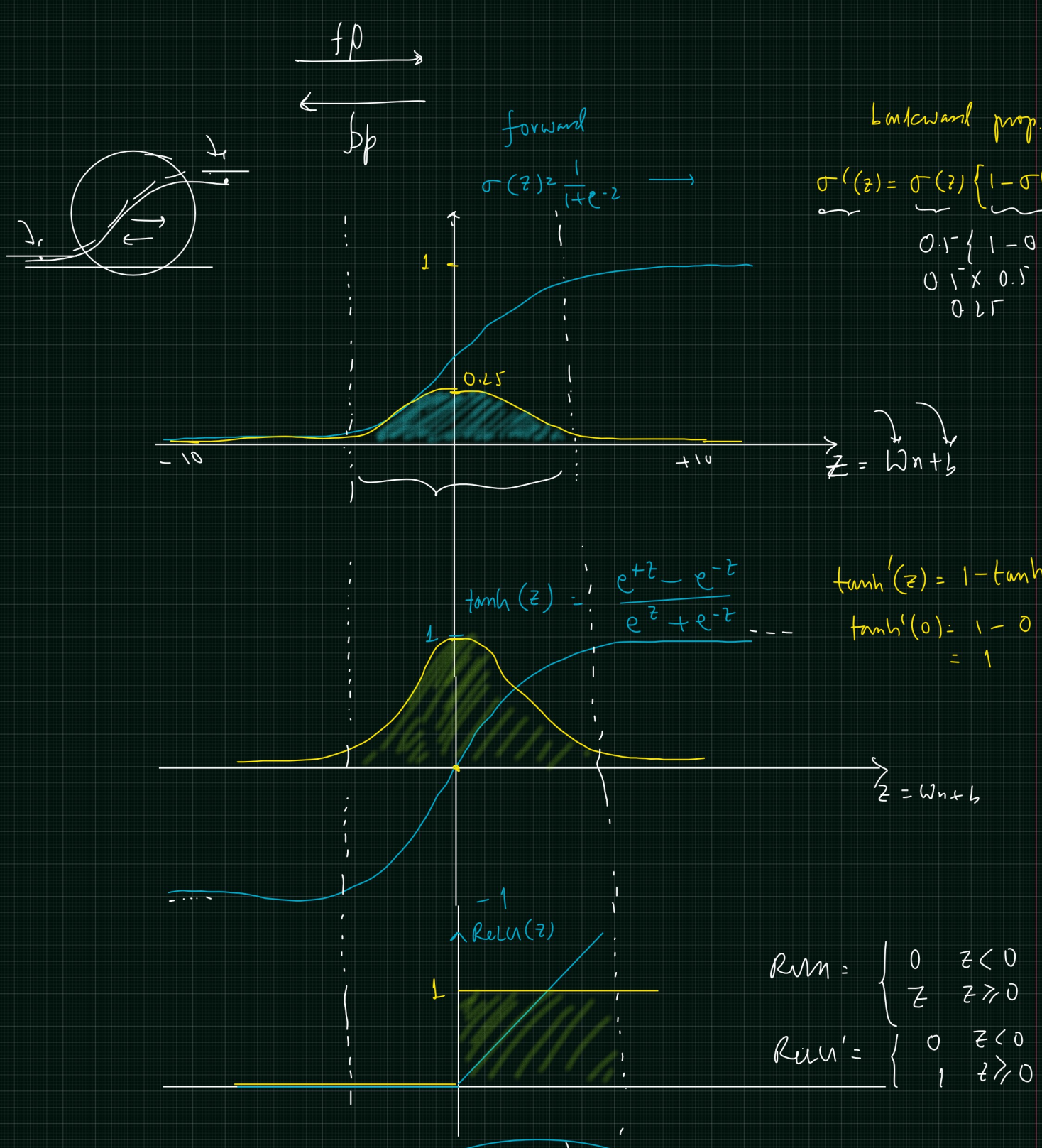
$z = 0$

$$\sigma(z=0) = \frac{1}{1+e^{0}} = \frac{1}{2} = 0.5$$

$$\square'(z=0) = \sigma(z=0)\{1-\sigma(z=0)\}$$

$$= 0.5(1-0.5)$$

$$= 0.25 \neq 0$$


fp
bp
forward
backward prop.
$\sigma(z) = \frac{1}{1+e^{-z}}$
$\sigma'(z) = \sigma(z)\{1-\sigma(z)\}$
0.1 { 1 - 0.5 }
0.1 x 0.5
0.25
1
0.25
-10
+10
$z = wx+b$
$\tanh(z) = \frac{e^{+z} - e^{-z}}{e^{z} + e^{-z}}$
$\tanh'(z) = 1 - \tanh^2(z)$
$\tanh'(0) = 1 - 0$
$= 1$
1
$z = wx+b$
-1
ReLU(z)
1
$\text{ReLU} = \begin{cases} 0 & z<0 \\ z & z \geq 0 \end{cases}$
$\text{ReLU}' = \begin{cases} 0 & z<0 \\ 1 & z \geq 0 \end{cases}$
$\sigma(z)$
$\hat{y} = \sigma(wx+b)$
$z$
$\sigma(wx+b)$
x
w
b
$z = wx+b$
$\frac{\partial e}{\partial w} = \frac{\partial e}{\partial \sigma} \cdot \frac{\partial \sigma}{\partial z} \cdot \frac{\partial z}{\partial w}$ - ①
$\frac{\partial e}{\partial b} = \frac{\partial e}{\partial \sigma} \cdot \frac{\partial \sigma}{\partial z} \cdot \frac{\partial z}{\partial b}$ - ②
$w = w - \eta \partial e/\partial w$
$b = b - \eta \partial e/\partial b$

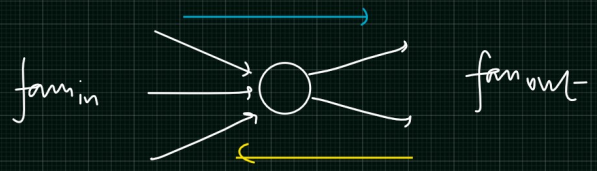
As per the paper,

for proper flow of information in forward as well as backward direction :-

info ≠ 0

$fan_{in}$ → no. of incoming edges to a layer

$fan_{out}$ → " " outgoing edges from ↓

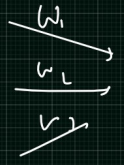



Condition to prevent vanishing & exploding gradient

$$fan_{in} = fan_{out}$$

alternative proposal

if $\sigma^2_{in} = \sigma^2_{out}$ ⇒ This can help you to maintain forward & backward info flow




↓

fanks the vanishing & exploding gradient issues

Calculate :-

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

For sigmoid,

Glorot initialization or Xavier initialization

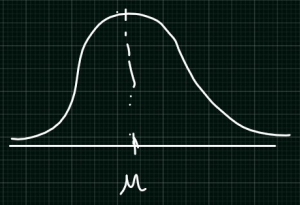
→ Normal distribution with $\mu = 0$ & variance $= \frac{1}{fan_{avg}}$

→ Uniform distribution between $-r$ & $r$ where $r = \sqrt{\frac{3}{fan_{avg}}}$

↳ Default case of weight initialization in Keras

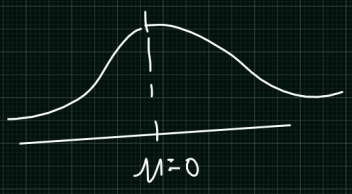
Normal distribution

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




Std Normal distribution $\mu = 0$, $\sigma = 1$

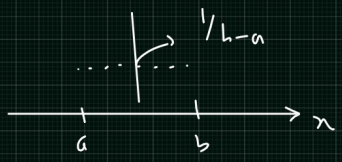
$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$




Uniform distribution

$a, b$

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



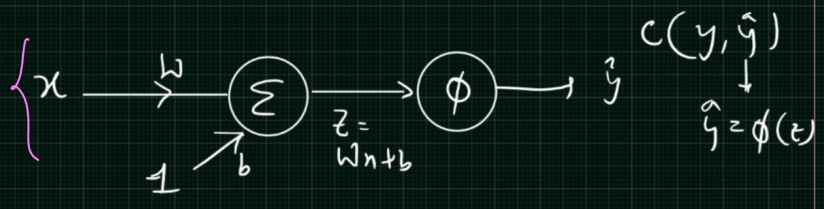

| | Initialization | Regression | $\sigma^2$ (normal) |
|---|---|---|---|
| Xavier Glorot & Yoshua Bengio | Glorot- | None, tanh, sigmoid, softmax | $1/fan_{avg}$ |
| Kaiming He | He | ReLU & its variants | $2/fan_{in}$ |
| Yann Lecun | Le Cun | SELU | $1/fan_{in}$ |

activation fn

tf.keras.layers.Dense(unit, activation = "relu", kernel_initialization = "he_normal")

$W_1 \ldots W_{100}$ $\longrightarrow$ $\sigma^2 = 2/fan_{in}$

# Activation Functions




forward pass ✓

$$\hat{y} = \phi(z)$$

where $z = wx + b$

domain → $(-\infty, \infty)$

range → $(0, 1)$ sigmoid

for forward pass :-

use $\phi(z)$

consider its domain & range

$\phi \rightarrow$ activation fn

Backward pass

$$\frac{\partial C}{\partial w} = \frac{\partial C}{\partial a} \cdot \frac{\partial a}{\partial z} \cdot \frac{\partial z}{\partial w}$$

$a' = \phi'(z) = \frac{\partial \phi}{\partial z}$ ✓

domain → $(-\infty, \infty)$ ←

range → $(0, 0.25)$ ←

sigmoid

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

at $z = 0$ $\sigma'(z=0) = 0.25$

$z = +\infty$ or $-\infty$ $\sigma'(z) \simeq 0$

for backward pass

use $\phi'(z)$

Consider its domain & range.

# 1) Sigmoid activation fn

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

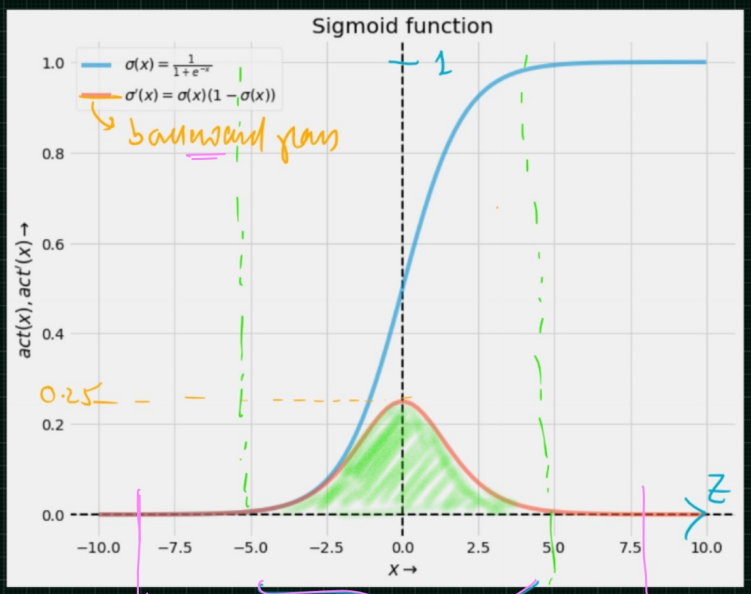
range $\sigma(z) \in (0, 1)$

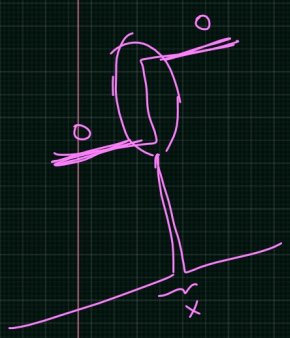
domain $z \in (-\infty, \infty)$

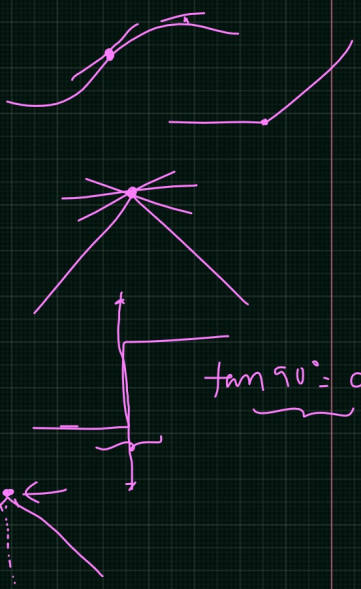
$$\sigma'(z) = \sigma(z)\{1-\sigma(z)\}$$

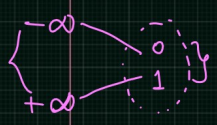
range $\sigma'(z) \in (0, 0.25)$

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






## Advantages :-

1) It has smooth gradient => prevents "jumps" in output.
   Continuous $f^n$ => derivable everywhere.
2) output value or Range $\sigma(z)$ is between 0 & 1.
   => normalizing the o/p of each neuron
   => help us to reduce the solution space.

## Disadvantages :-

1) Prone to gradient vanishing
   $\sigma'(z) \in (0, 0.25)$ $\quad 0.25 < 1$
2) Power operation or exponential terms makes calculation time consuming
   ⇓
   latency ↑

$\left\{\frac{1}{1+e^{-z}}\right\}$

# 2. Hyperbolic Tangent activation function

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

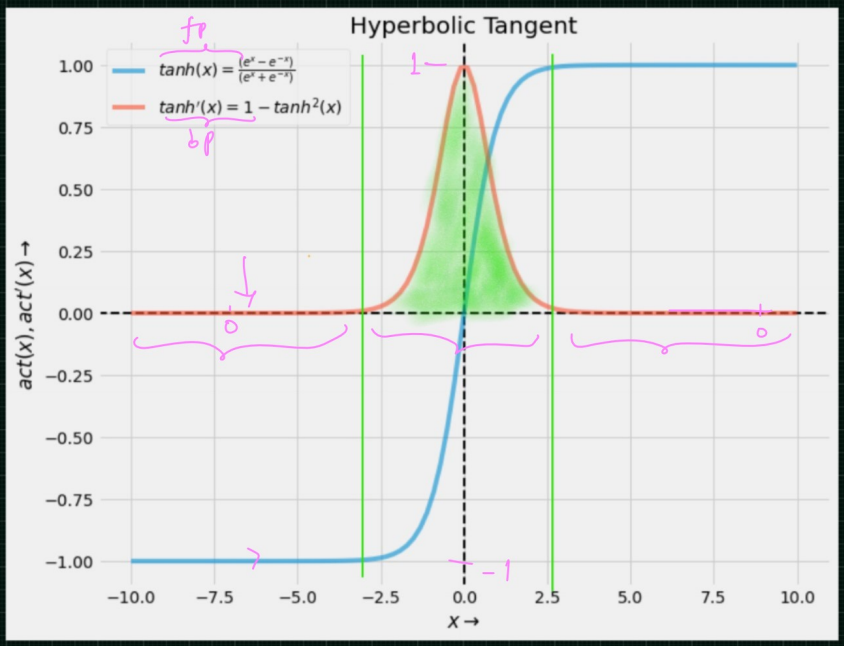
range → $\tanh(z) \in (-1, 1)$

domain → $z \in (-\infty, \infty)$

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

range → $\tanh'(z) \in (0, 1)$

domain → $z \in (-\infty, \infty)$




(0, 1)

Advantages:-

1) Smooth gradients.
2) Symmetric to origin ⇒ Zero mean
3) Solves vanishing gradient issue to some extent provided there is a proper Wt. initialization and also because its range for $\tanh'(x)$ is between (0, 1)
4) Suitable function for hidden layers

Disadvantage.

1) It also has saturation region
2) It introduces latency because of exponential terms.

## 3> Rectified linear unit (ReLU) activation fn.

$$ReLU(z) = max(z, 0)$$

— or —

$$ReLU(z) = \begin{cases} z & z \geq 0 \\ 0 & z < 0 \end{cases}$$ if else

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

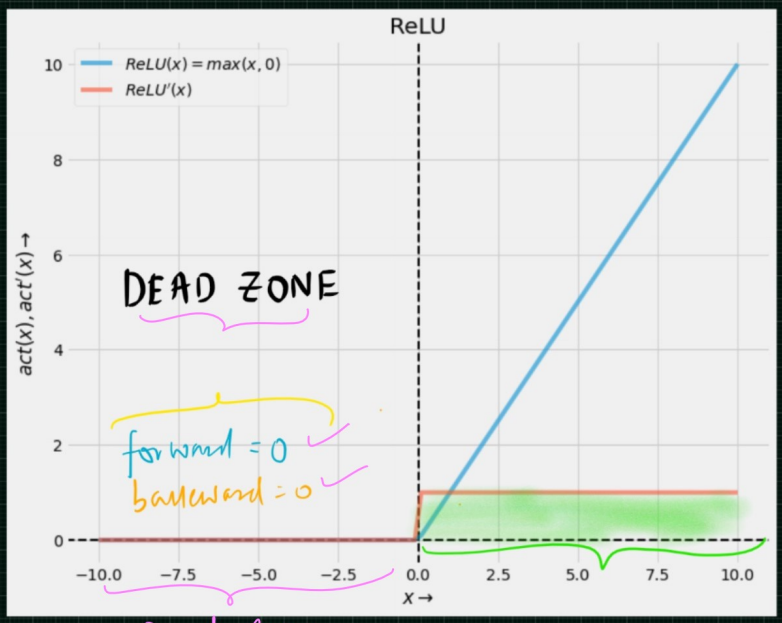
range $ReLU(z) \in [0, \infty)$

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

derivative is not defined at $z = 0$

∴ domain $z \in (-\infty, \infty)$ except $z = 0$

range $ReLU'(z) \in 0$ and $1$.




Dead ReLU

### Advantages :-

1> If input is +ve => There is no gradient saturation problem

2> Calculation is much faster {forward or backward} as compared to tanh & sigmoid.

### Disadvantage :-

1> for -ve inputs ReLU is completely inactive

⇓

Dying ReLU problem.

2> Not a zero centric function.

3> At zero its derivative is not defined => jump

# 4) Leaky ReLU function

$\frac{d(\alpha z)}{dz} = \alpha(1) = \alpha$

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

— or —

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

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

Range of LeakyReLU$(z) \in (-\infty, \infty)$

$\alpha = 0.01$ (0.01z)

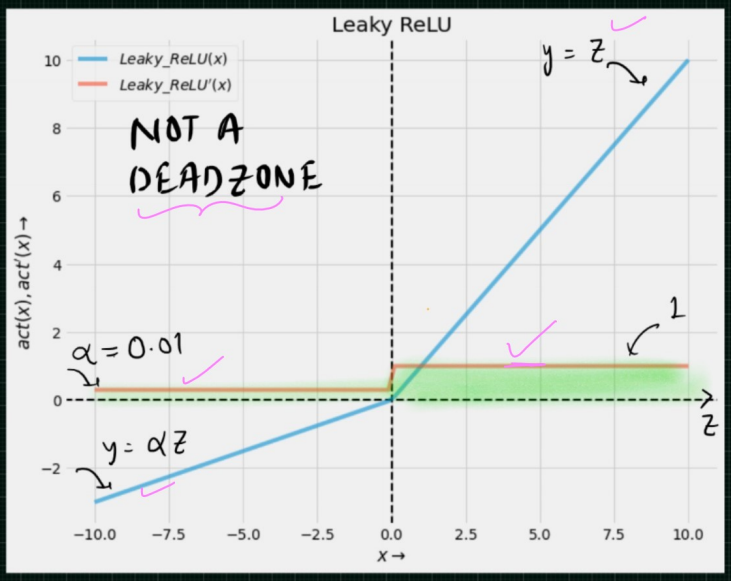
$$\text{Leaky ReLU}'(z) = \begin{cases} 1 & z > 0 \\ \alpha & z < 0 \end{cases}$$

derivative is not defined at $z = 0$

∴ domain $z \in (-\infty, \infty)$ except $z = 0$

range: 0.01 or 1

Leaky ReLU$'(z) \in \alpha$ or $1$.




## Advantages:-

1) ~~If input is +ve~~ for all input ⇒ There is no gradient saturation problem

2) Calculation is much faster {forward or backward} as compared to tanh & sigmoid.

## Disadvantage:-

1) ~~for -ve inputs ReLU is completely inactive~~

⇓

~~Dying ReLU problem.~~

2) Not a zero centric function.

3) At zero its derivative is not defined ⇒ jump

# 5> PRelu (Parametric ReLU)

$$PRelu(z) = \begin{cases} z & z \geq 0 \\ \alpha z & z < 0 \end{cases}$$

$\alpha$ → learnable parameter / trainable

$$\omega = \omega - \eta \frac{\partial e}{\partial \omega}$$

$$\alpha = \alpha - \eta \frac{\partial e}{\partial \alpha}$$

if $\alpha = 0$ ⇒ ReLU

if $\alpha > 0$ ⇒ Leaky ReLU

if $\alpha$ is learnable parameter ⇒ PRelU

**Advantage:**

1> $\alpha$ is now intelligently updated / learnt during the training

⇓

No manual tweaking required to set $\alpha$

⇓

$\alpha$ will be adaptable as per the data

**Disadvantage:**

1> You have extra parameter $\alpha$ to train

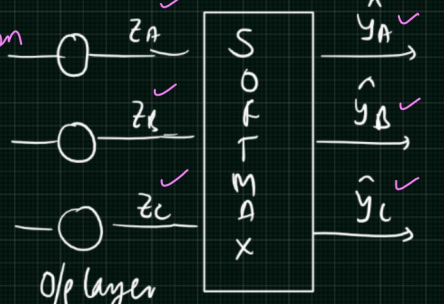
⇓

Training time ↑

# SOFTMAX ACTIVATION FUNCTION:-

no. of classes $\geq 2$

At o/p layer multiclass classification




$$\Sigma e = e^{z_A} + e^{z_B} + e^{z_C} \quad -(1)$$

$$\frac{e^{z_A}}{e^{z_A} + e^{z_B} + e^{z_C}}$$

$$\hat{y}_A + \hat{y}_B + \hat{y}_C = \frac{e^{z_A} + e^{z_B} + e^{z_C}}{\Sigma e} = \frac{\Sigma e}{\Sigma e} = 1$$

Outcome: Probability distribution

$$\Sigma y = 1$$

$\hat{y}_A = 0.7 \rightarrow$ 70% of chance is the prediction belongs to class A

$\hat{y}_B = 0.3 \rightarrow$

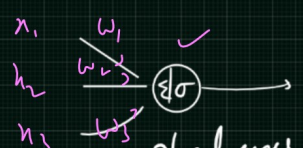
$\hat{y}_C = 0.0$

Case Study:-

Case 1: Binary classification. { sigmoid }

60% $0.6 \rightarrow$ A

$0 \rightarrow 0.5$ A, $0.5 \rightarrow 1$ B




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

$$z = w_1x_1 + w_2x_2 + w_3x_3 = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}^T \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$w^T \cdot x = \theta^T \cdot x$$

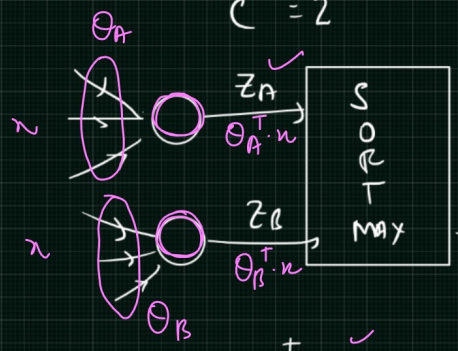
$$(w_1\ w_2\ w_3) \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

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

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

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

Case 2: act fn: softmax Binary classification




$$\hat{y}_A \quad \frac{e^{z_A}}{e^{z_A} + e^{z_B}} = p(y=1|x) = \frac{e^{\theta_A^T \cdot x}}{e^{\theta_A^T \cdot x} + e^{\theta_B^T \cdot x}}$$

$$\hat{y}_B \quad \frac{e^{z_B}}{e^{z_A} + e^{z_B}} \quad \therefore p(y=0|x) =$$

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

$$= \frac{1}{1 + e^{\theta_B^T \cdot x - \theta_A^T \cdot x}}$$

$$= \frac{1}{1 + e^{-(\theta_A^T - \theta_B^T) x}}$$

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

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

$$p(y=0|x) = \frac{e^{\theta_B^T \cdot x}}{e^{\theta_A^T \cdot x} + e^{\theta_B^T \cdot x}}$$

$$= \frac{e^{\theta_B^T \cdot x - \theta_A^T \cdot x}}{1 + e^{\theta_B^T x - \theta_A^T \cdot x}}$$

$$= \frac{e^{-(\theta_A^T - \theta_B^T) x}}{1 + e^{-(\theta_A^T - \theta_B^T) x}}$$

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

Conclusion: For binary classification Softmax is equivalent to sigmoid

Drawback :-

i) It contains exponential terms ✓

⇓

computationally intensive ✓

Softmax is always used in output layer

## 2) ELU ( Exponential Linear Units )

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

forward pass

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

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

$\alpha$ = scalar value | it's a slope of -ve section

at $z = 0$, let's see continuity of elu ✓

$\xrightarrow{0^-} \; 0 \; \xleftarrow{0^+}$

$$\begin{cases} LHL = \lim\limits_{z\to 0^-} elu(z,\alpha) = \lim\limits_{z\to 0^-} \alpha(e^z - 1) \simeq \alpha(e^0 - 1) \\ \quad = \alpha(1-1) = 0 \end{cases}$$ ✓

$$\begin{cases} RHL = \lim\limits_{z\to 0^+} elu(z,\alpha) = \lim\limits_{z\to 0^+} z = 0 \end{cases}$$ ✓

$elu(z=0,\alpha) = 0$

$\therefore$ LHL = RHL = $elu(z=0,\alpha)$

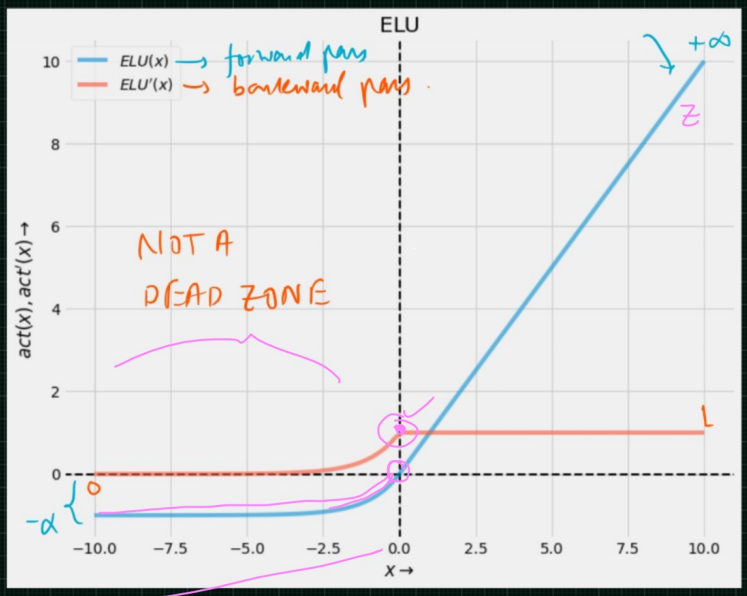
$\Rightarrow$ elu is continuous. ✓

Backward pass

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

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

range $elu'(z,\alpha) \in (0, 1]$




**Advantage :-**

i) No dead ReLU issues
ii) Approx zero centered

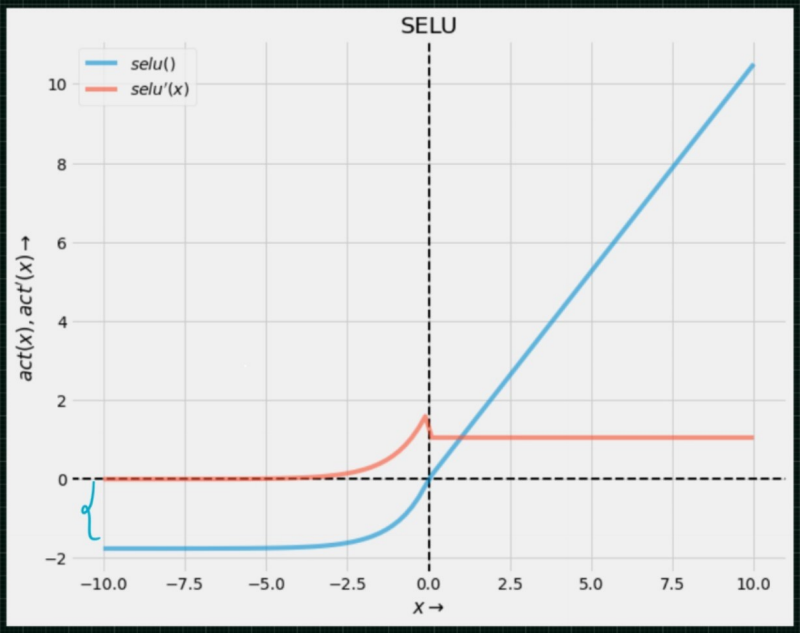
**Disadvantage :-**

i) exponential term ⇒ compute intensive

## 3) SELU → Scaled ELU

$$selu(z, \alpha) = scale \times elu(z, \alpha)$$

$$= k \times elu(z, \alpha)$$



## 4) Swish

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

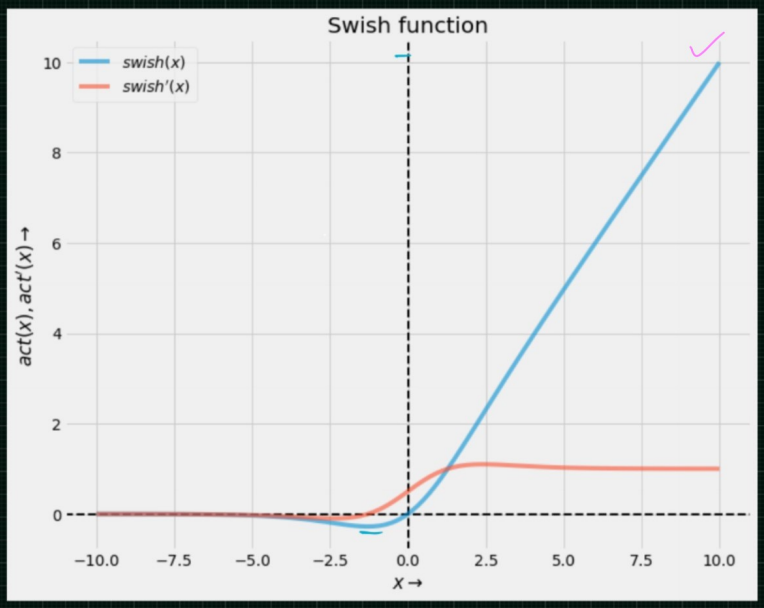
domain $z \in (-\infty, \infty)$

range $\in (0, \infty)$

$\beta$ can be a learnable parameter

for $\beta = 1$.

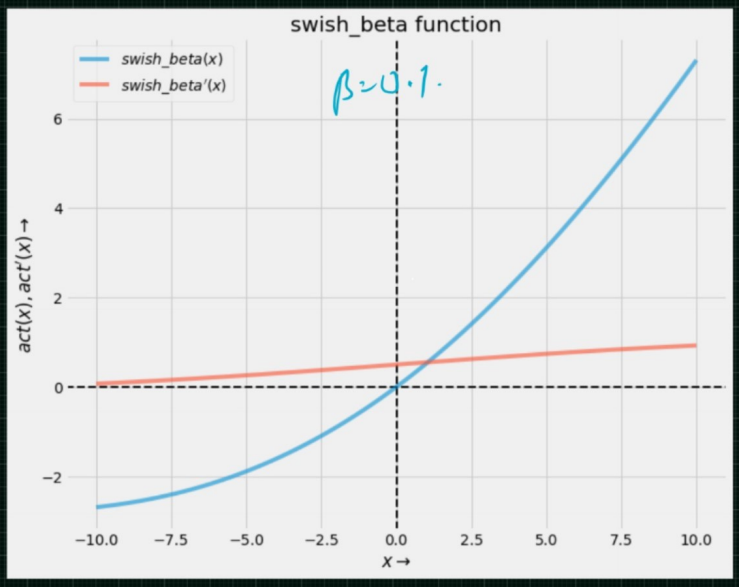
$$\text{swish} - 1 \Rightarrow \text{swish}(z) = z \cdot \sigma(z) = \frac{z}{1 + e^{-z}}$$




derivative of swish

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

range $\in (0, 1)$




Drawbacks :-

i) Presence of $e^x$ term

↓

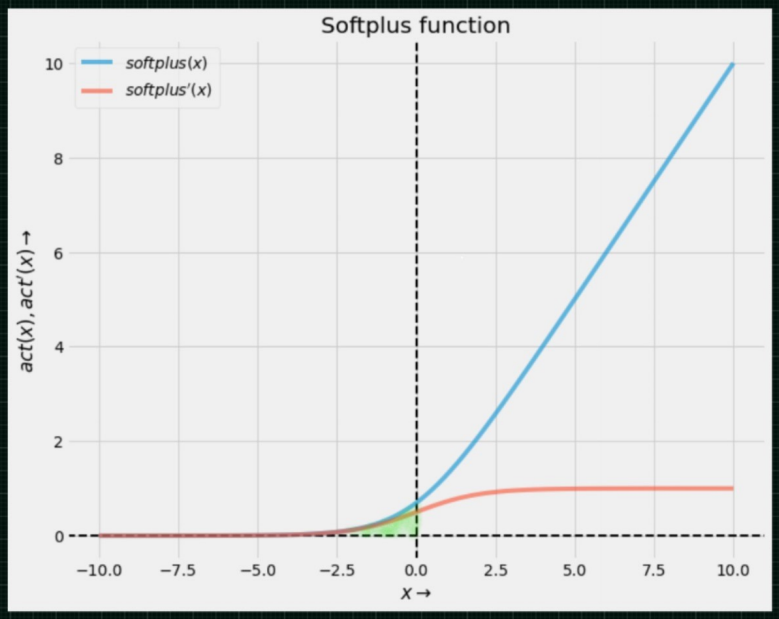
compute intensive

Advantages :-

i) Its unique the origin

# SOFTPLUS

$$\text{softplus}(z) = \log(1 + e^z)$$




**Disadvantage :-**

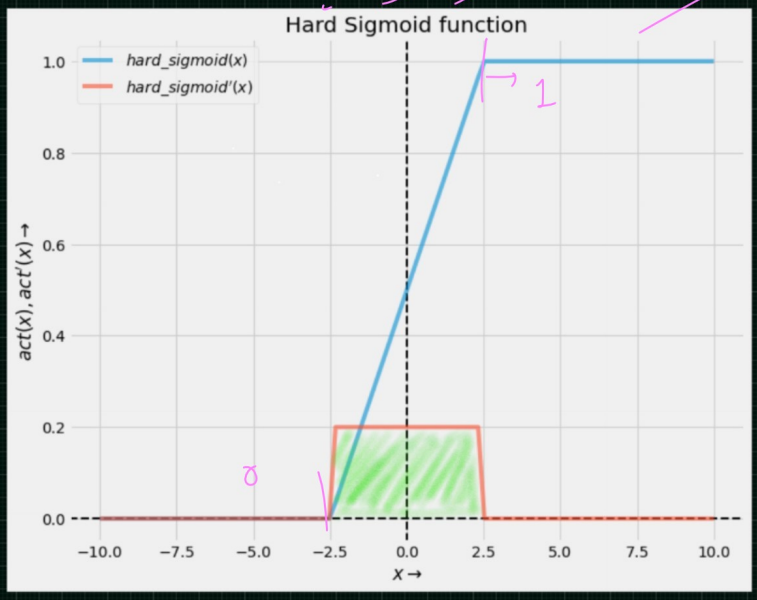
i) Presence of $e^z$ & log ⇒ computation intensive

**Advantage :-**

i) Is differentiable everywhere

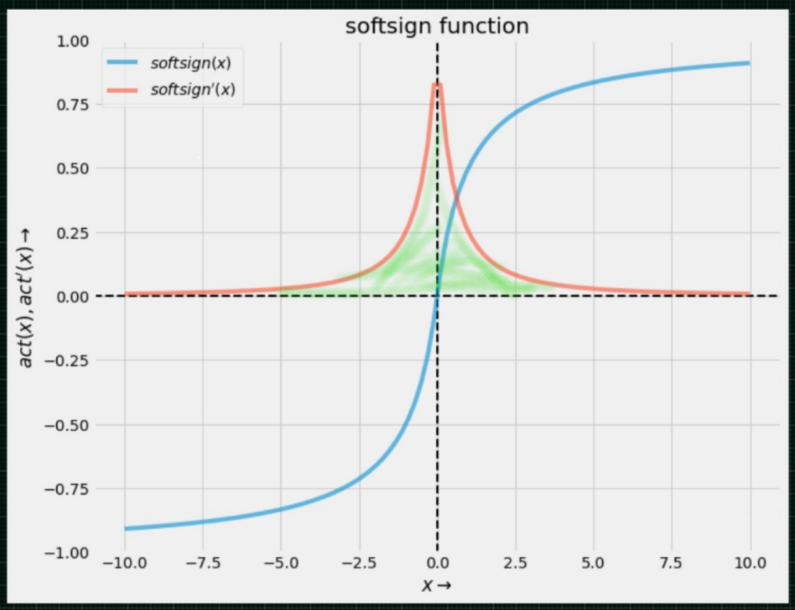
# Hard Sigmoid

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

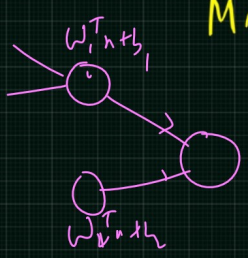


## SOFTSIGN

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



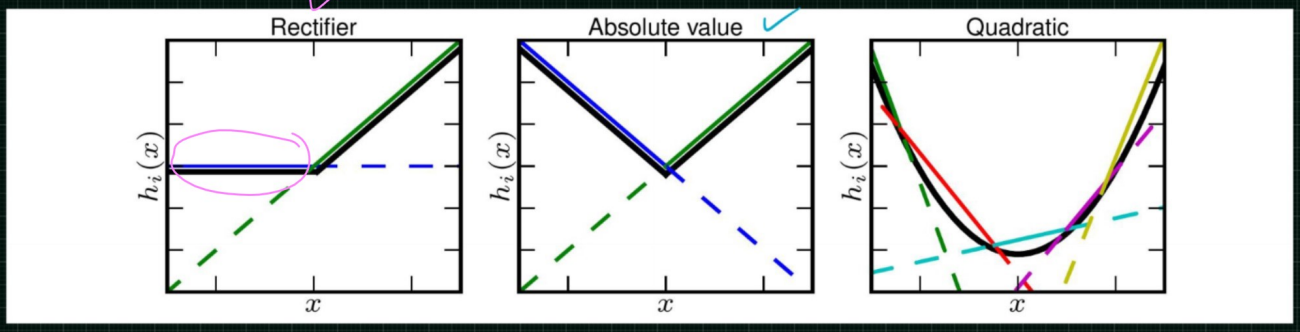

## MAXOUT




Generalization of ReLU & Leaky ReLU

$$f(x) = \max\left(W_1^T x + b_1, W_2^T x + b_2\right)$$

$$\left(0, (\ )\right)$$




Drawback :-

i) It is computationally intensive as it doubles the number of parameters.

→ How to start / begin selection of activation function?

for any problem,

Go with ReLU for hidden layers (90%)

hid — ReLU, hid variant ReLU (Leaky ReLU, Parametric ReLU, ELU)

o/p — softmax

versions of relu { as a future trend }

for classification :-

binary :-

sigmoid / softmax at o/p layer

o/p unit = 1 o/p unit = 2

Multiclassification

softmax only at o/p layer

for Regression :- o/p

No activation function

Remark

DL → Architecture → CNN → Std data

Imagenet, CIFAR } previous