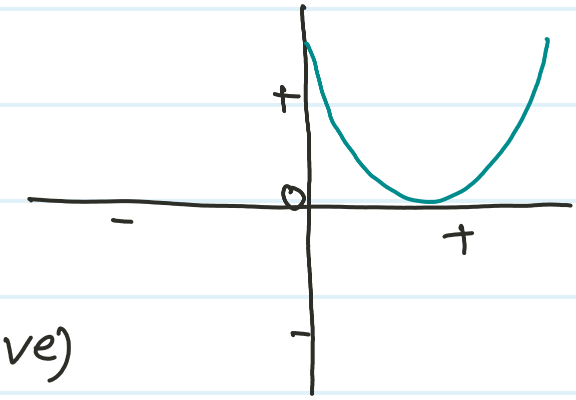
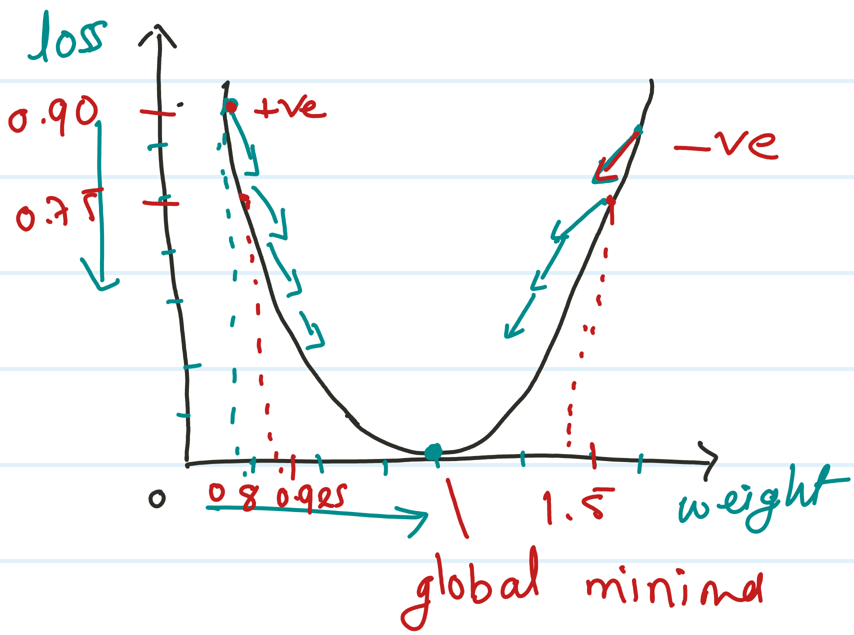
optimizer — use to reduce loss and update weight

⇒ Gradient optimizer

$$w_{new} = w_{old} - \eta \left(\frac{\partial E}{\partial w_{old}} \right)$$



$$= \frac{\partial E}{\partial w_{old}} = (+ve) \text{ or } (-ve)$$



$$y - \hat{y} = 0.75$$

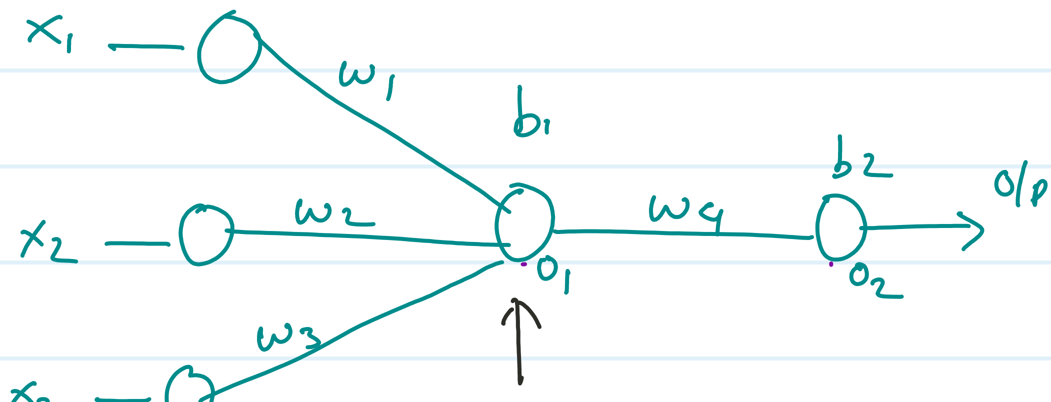
$$w_{new} = 0.8 + 0.5 \times (0.25)$$

$$= 0.8 + 0.125 = 0.925$$

$$w_{new} = 1.5 + 0.5(-0\,2)$$

$$= 1.5 - 0.10 = 1.4$$

chain rule of Derivative



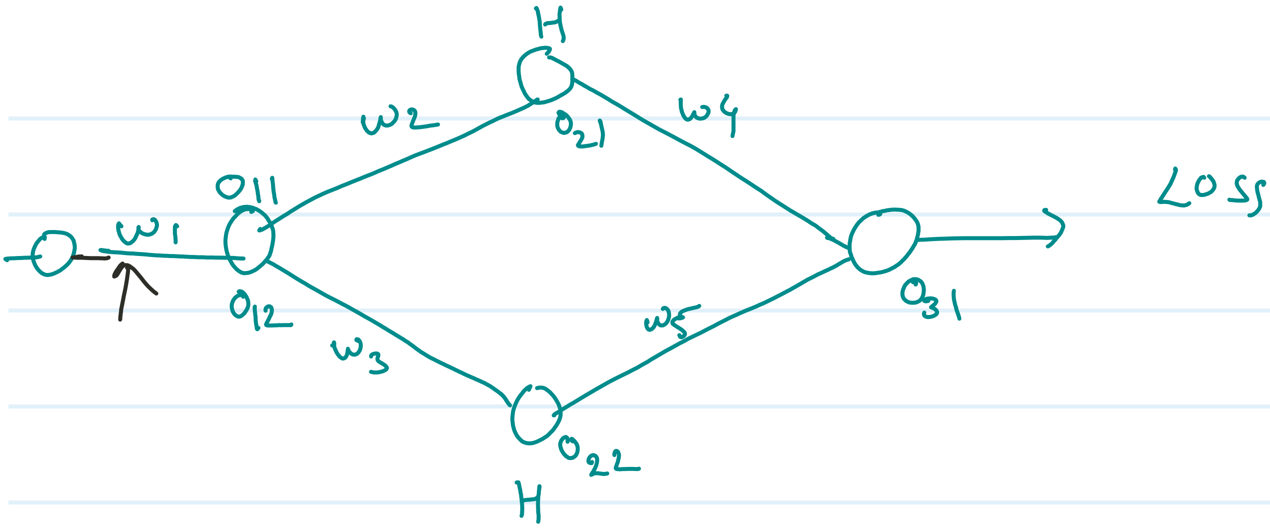
loss $(y-\hat{y})^2$

$$W_{4new} = W_{4old} - \eta \frac{\partial E}{\partial w_{4old}} \quad \text{— slop}$$

$$\frac{\partial E}{\partial w_{4old}} = \frac{\partial E}{\partial O_2} \times \frac{\partial O_2}{\partial w_{4old}}$$

$$w_{1new} = w_{1old} - \eta \frac{\partial E}{\partial w_{1old}}$$

$$\frac{\partial E}{\partial w_{1old}} = \frac{\partial E}{\partial O_2} \times \frac{\partial O_2}{\partial O_1} \times \frac{\partial O_1}{\partial w_{1old}}$$

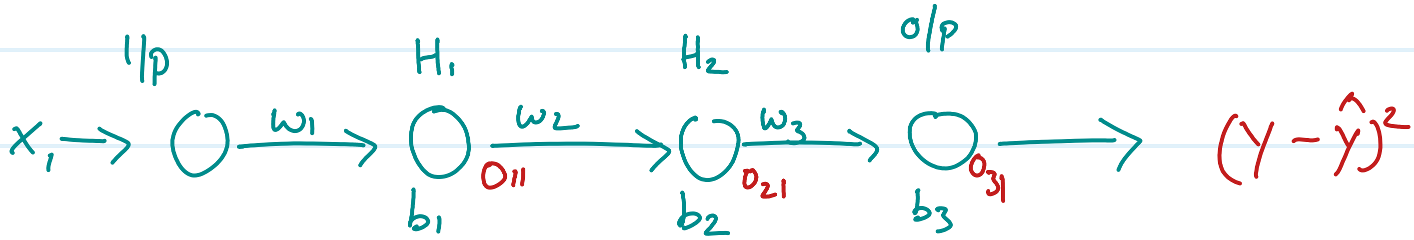


$$w_{1_{new}} = w_{1_{old}} - \eta \frac{\partial \varepsilon}{\partial w_{1_{old}}}$$

$$\frac{\partial \varepsilon}{\partial w_{1_{old}}} = \left[\frac{\partial \varepsilon}{\partial O_{31}} \times \frac{\partial O_{31}}{\partial O_{21}} \times \frac{\partial O_{21}}{\partial O_{11}} \times \frac{\partial O_{11}}{\partial w_{1_{old}}} \right] +$$

$$\left[\frac{\partial \varepsilon}{\partial O_{31}} \times \frac{\partial O_{31}}{\partial O_{22}} \times \frac{\partial O_{22}}{\partial O_{12}} \times \frac{\partial O_{12}}{\partial w_{1_{old}}} \right]$$

* Vanishing Gradient Problem and Activation Function



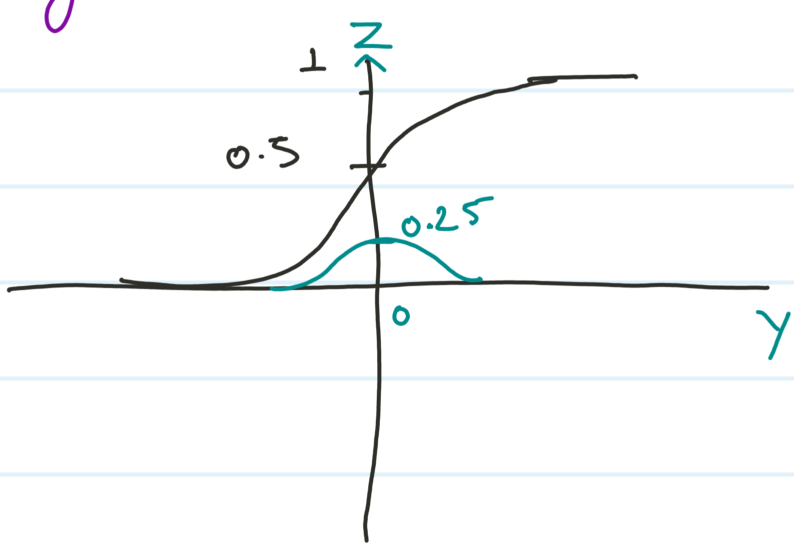
Sigmoid funct -

It gives of value b/w 1 to 0.

Forward propogation

$$\sigma(z) = \frac{1}{1+e^{-z}} = [0 \text{ to } 1]$$

whenever we find derivative so sigmoid fun. will range b/w $0 \leq \sigma(z) \leq 0.25$



$$W_{new} = W_{old} - \eta \frac{d\mathcal{E}}{\partial w_{old}}$$

$$\frac{\partial O_{31}}{\partial O_{21}} = [0 - 0.25]$$

$$\Rightarrow \overset{1}{0.25} \times \overset{2}{0.15} \times \overset{3}{0.10} \times \overset{4}{0.05}$$

Suppose we are getting low value, we get small

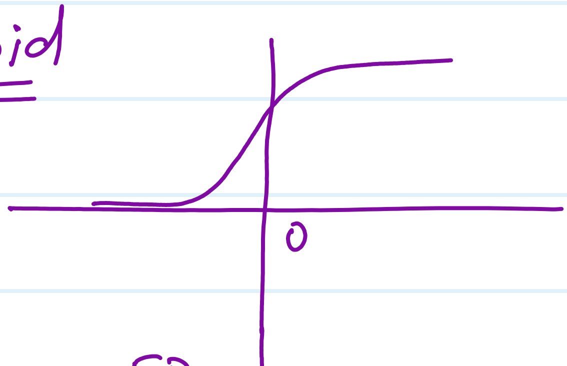
$$w_{new} = w_{old} - \eta(\text{small})$$

$$w_{new} = w_{old}$$

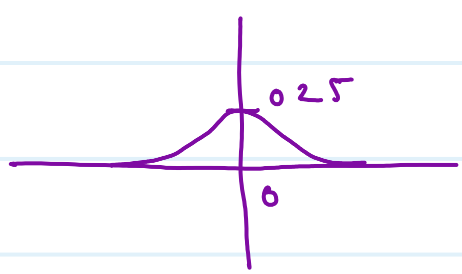
This is a problem of vanishing gradient.

To fix this problem we use other Activation func. ① Tanh ② ReLU ③ PrReLU ④ Leaky ReLU ⑤ ELU ⑥ softmax

Sigmoid



FP



BP

since in the sigmoid fun. curve is not becoming zero centric.

Advantage

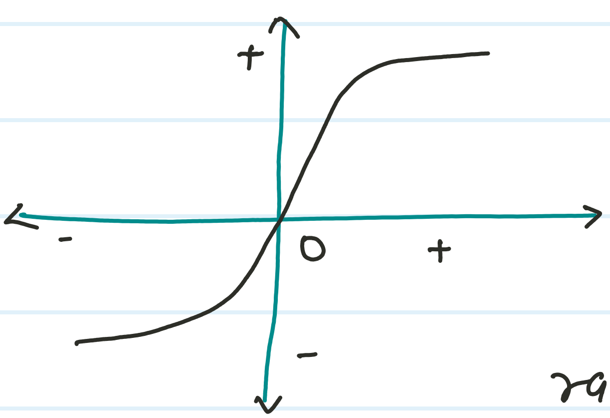
clear 0/1

Disadvantage

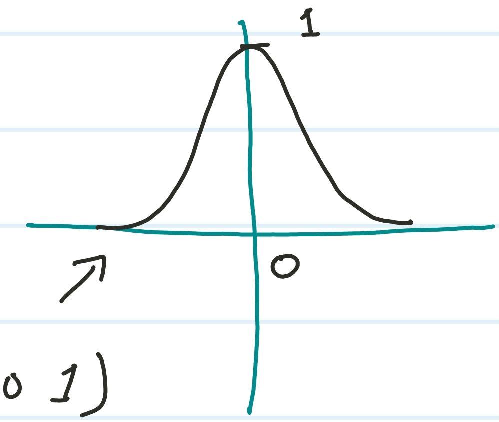
① prone to vanishing Gradient
② It is not zero centric
③ Time complexity

Tanh –

FP



BP



range (0 to 1)

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \qquad = \frac{\partial(\tanh(x))}{\partial x}$$

range = -1 to 1

Adv.

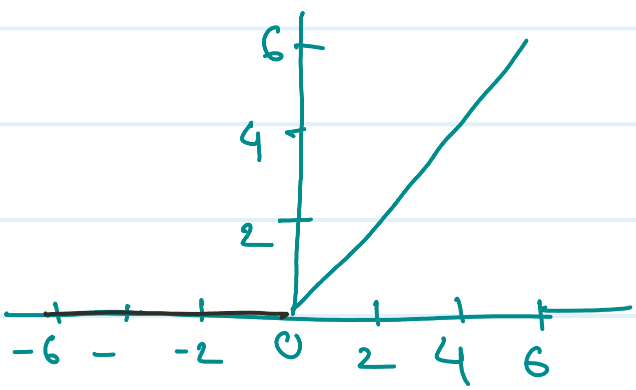
① It is zero centric

Disadv.

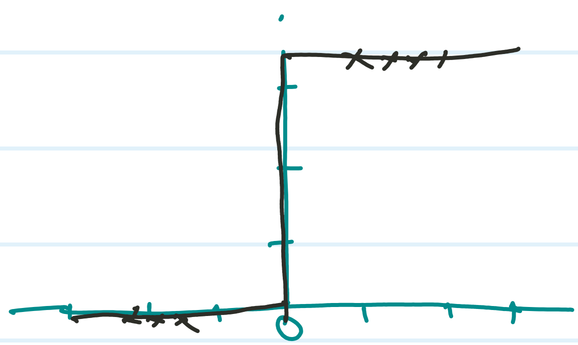
① Time complexity
② vanishing gradient proble
③ Not perfect for deep neural network

ReLU (Rectified Linear unit)

FP ·



BP



(0 to 1)

$$\frac{\partial(\text{ReLU}(x))}{\partial x}$$

$$\text{ReLU} = \max(0, x)$$

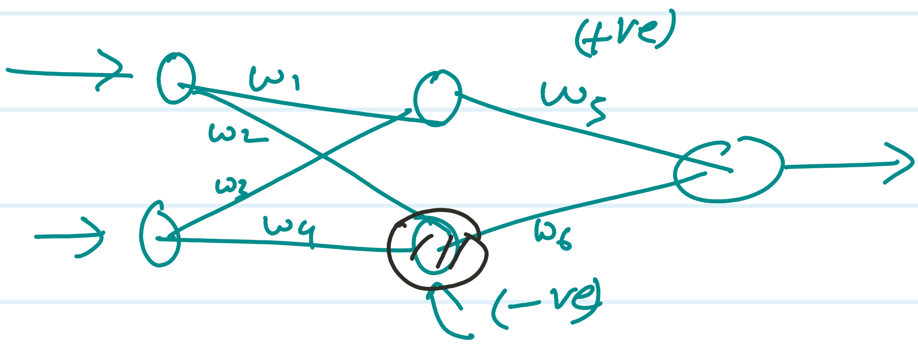
Adv.

① It is faster than Tanh and sigmoid

Disadv.

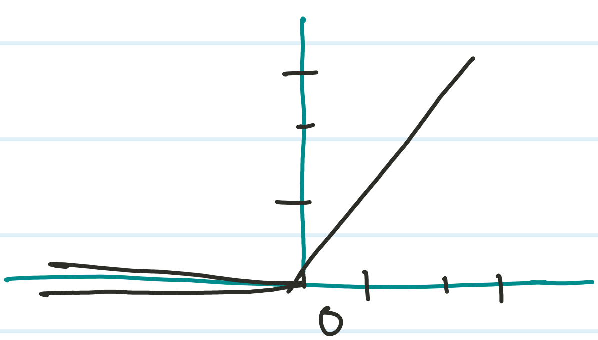
① It create dead neuron

If any weight -ve, so one of the neuron become dead it will be affect entire chain rule

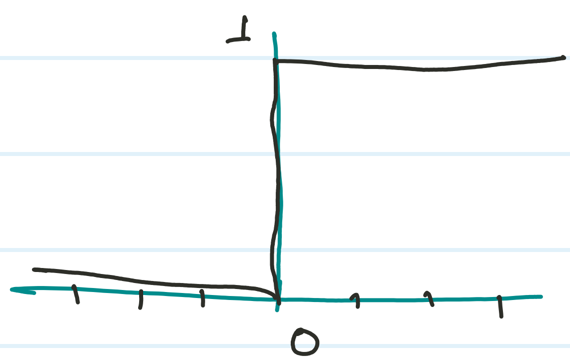


Leaky ReLU

FP



BP



$$\text{Leaky ReLU} = \max(0.01x, x)$$

Adv

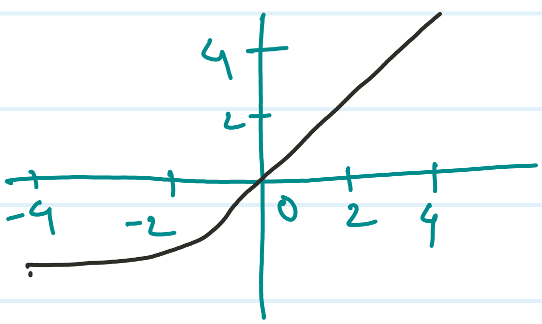
① prevent from dead neuron

Disadv:

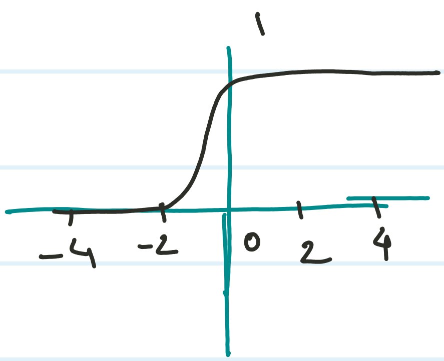
① It is not zero centric.

ELU (Eponential Linear units)

FP



BP



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

if $x = 0$ become ReLU

if $x > 0$ become Leaky ReLU

if a_i is lineaable parameter it become PreReLU

Adv

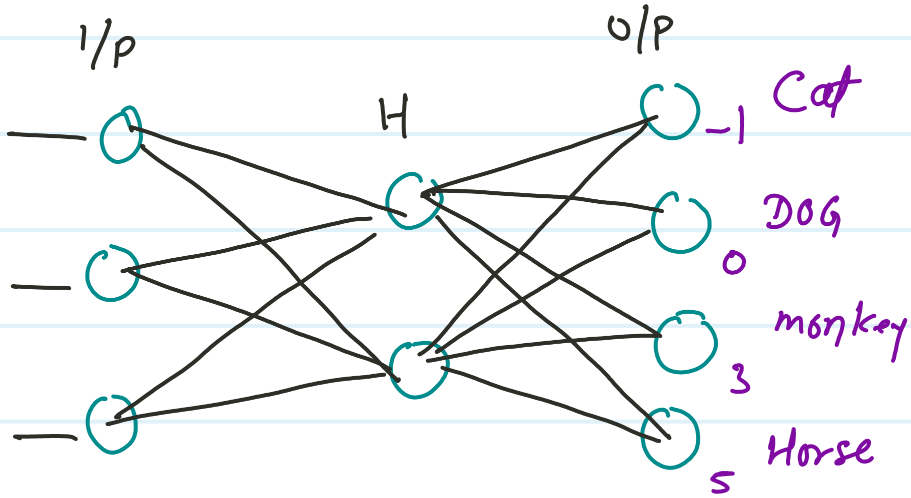
① zero centric

② prevent from dead neuron

Disadv.

① computation power is heavy or more.

Softmax Activation function



$$\text{softmax} = \frac{e^{y_i}}{\sum_{k=0}^{n} e^{y_k}}$$

Softmax Activation —

$$\text{cat} = \frac{e^{-1}}{e^{-1} + e^{0} + e^{3} + e^{5}} = 0.00033$$

$$\text{Dog} = \frac{e^{0}}{e^{-1} + e^{0} + e^{3} + e^{5}} = 0.1024$$

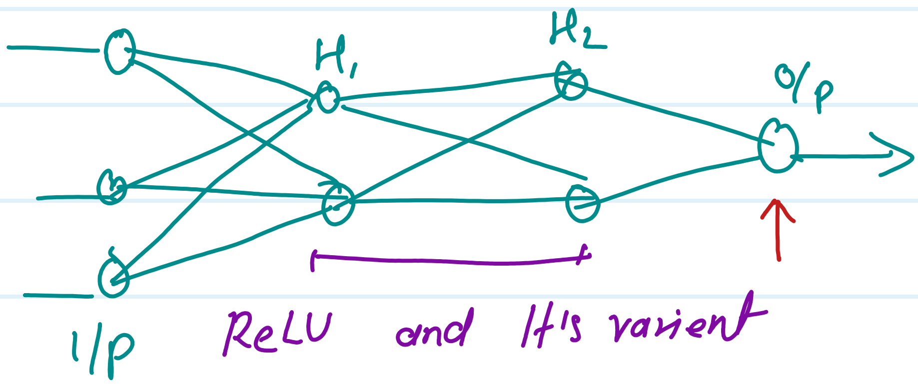
$$\text{monkey} = \frac{e^{3}}{e^{-1} + e^{0} + e^{3} + e^{5}} = 0.0183$$

$$\text{Horse} = \frac{e^{5}}{e^{-1} + e^{0} + e^{3} + e^{5}} = 0.1353$$

$$pr(Horse) = \frac{0.1353}{0.00035 + 0.024 + 0.0183 + 0.1353}$$

$$= 0.86 = \boxed{86\%}$$

Which activation function to use when



For Binary class / on output layer – Sigmoid funct.

for multiclass / on output layer – Softmax Funct.

In the both condition we'll use ReLU and it's varient on hidden layer.