Deep Learning

AI: "Application which can do its own task. without any human Intervention"

Ex: Netflix App, Self Dreving Cars.

Subset

ML: Statical tool do analize, visualize the data. Predictive, forecasting, clustering.

Subset

DL: To Mimic the human brain behaviour. And learn from it.

DS: A Part of AI, ML, DL

Why Deep learning?

Ans: 2005 → Facebook, insta, whatsapp. generated a Manive data which need to anlaysicl and. used for developing buisness. To mimic the human behavior and Memorise the data and Learning from data, deep learning is required.

Hardware Advancement: GPU's used to train the DL models.

Ex: RTx Titan

RTx 3090

CPU: has several cores, Serial processing. large Memory capacity. Small batch operation Ex: intel
= AmD

GPU :(Graphics processing unit) its. works as an performance accelerator for CPU. has thousands of cores. High throughputs, Parallel processing. Executing thousands of operations at once.

Ex: NVIDIA

AMD

Google

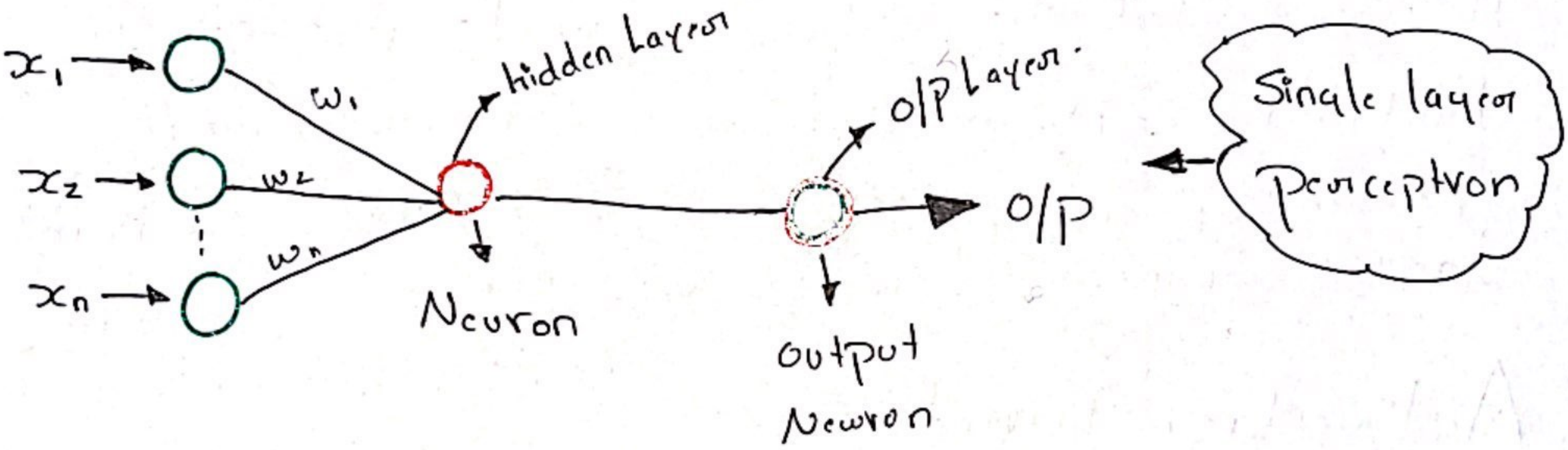
BROADCOM

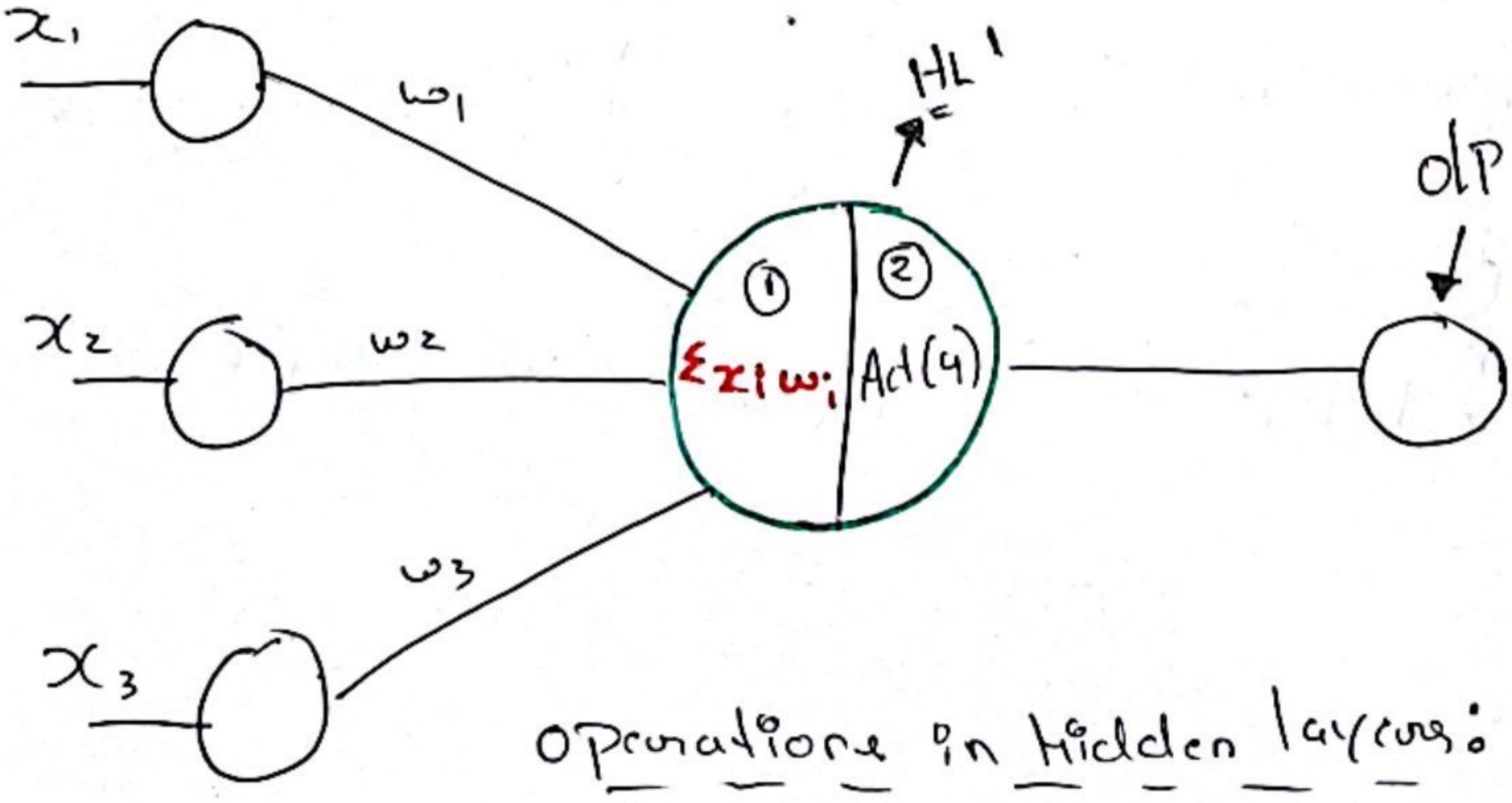
TPU : Tensor processing unit. Application Specific integrated circuits. fast at Performant dense vector & matrix. computation to accelerate neural netwo.

- very throughput ex: google
  high
- Extreme parallem. • coral
- Highly optimized for large batch • HAILO

Perceptron

Perceptron is also understood as an "Artificial Neuron" or "neural networks" unit that helps to detect certain input data computation in business intelligence.





Operations in Hidden layers: (Y) $= \Sigma x_i w_i + b$

① $\Sigma x_i w_i = x_1 w_1 + x_2 w_2 + x_3 w_3$ (Summation)

$= w^T x$

② Activation function.

Weight importance ?

Ans: which weight should have what value
So that the neuron should be activated
till that level. (More the weight to the
important more important to neuron).
neuron.

Bias ?

Each hidden layer has a bais

it is added to weights for to give certain
best value. if weight are 0 to overcome the
Problem of $x_i w_i$. Bias is added. (it is a constant)
added for product of feature and weight) to help to shift.
AF to positive or negative side.

Activation Function

based

it classify the output to 0 or 1 on the
the Act function (sigmoid). ($y = \sum x_i w_i + b$)

Sigmoid $\frac{1}{1+e^{-y}}$ - $\frac{1}{1+e^{-(\sum x_i w_i + b)}}$

$\geq 0.5 = 1$
$< 0.5 = 0$

Forward Propogation.

(1) features are the inputs. x_i

(2) Initilize the weights to each x_i corrsponit w_i

(3) Formation

Multiple the weights with x_i, add bias.

$$y = (\Sigma x_i w_i) + b$$

(4) Pass it to activation function

(5) output layer recived a value.

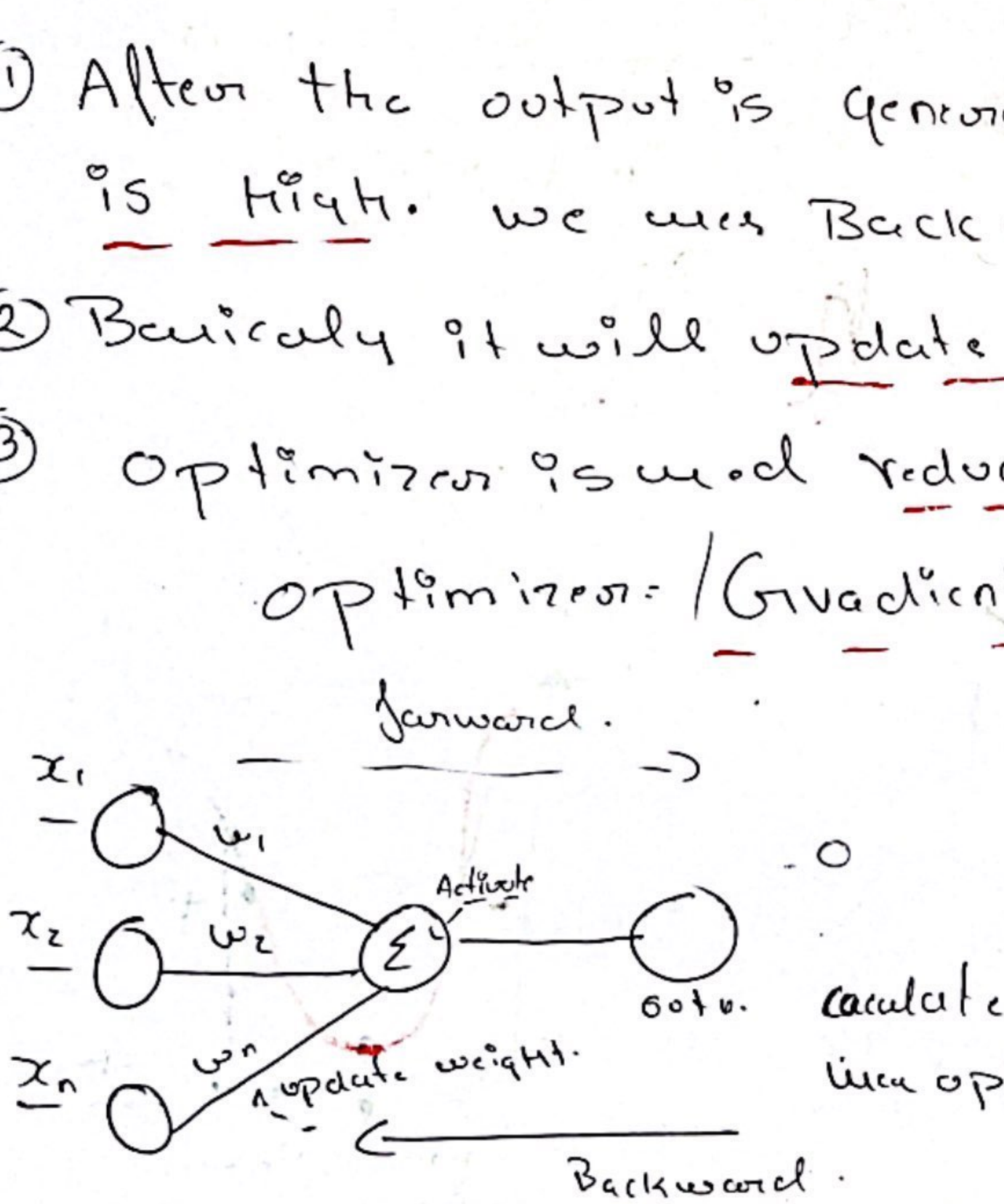
Back Propogation.

(1) After the output is generated if the loss function is High. we use Back propogation.

(2) Basicaly it will update the weights..

(3) optimizer is used reduce the loss function.

optimizer: (Gradient decient).



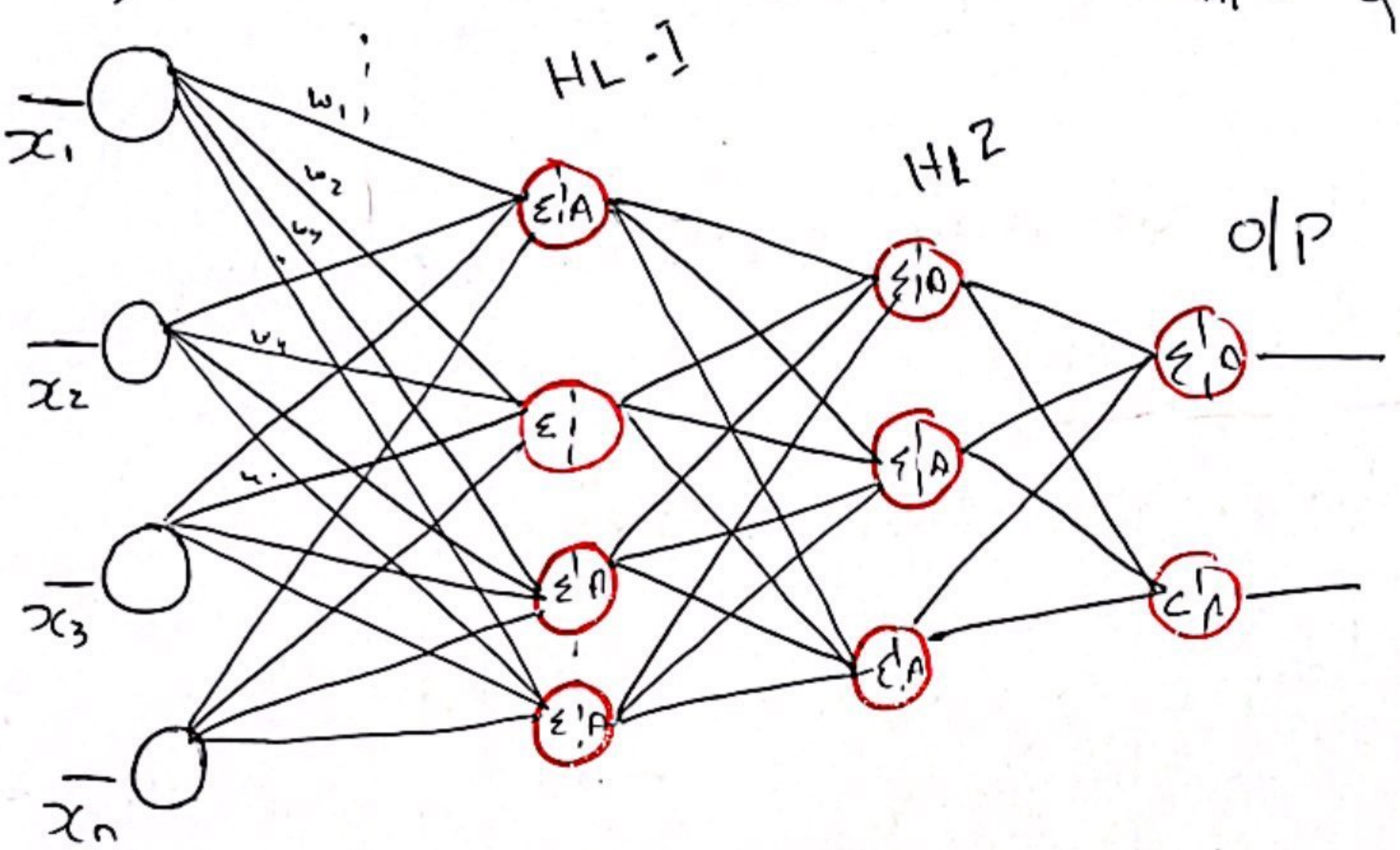
calculate a loss. if High use optimizer to reduce loss.

Multi Layer Neural network.

$$\Sigma = (\Sigma w_i x_i) + b$$

$$\text{用} = y = \frac{1}{1+e^{-y}}$$

$$y = \frac{1}{1+e^{-(\Sigma w_i x_i + b)}}$$



Back propagation or weight updation

Formula: GD

$$W_{new} = W_{old} - \eta \frac{\partial L}{\partial w_{old}}$$

η → learning rate.

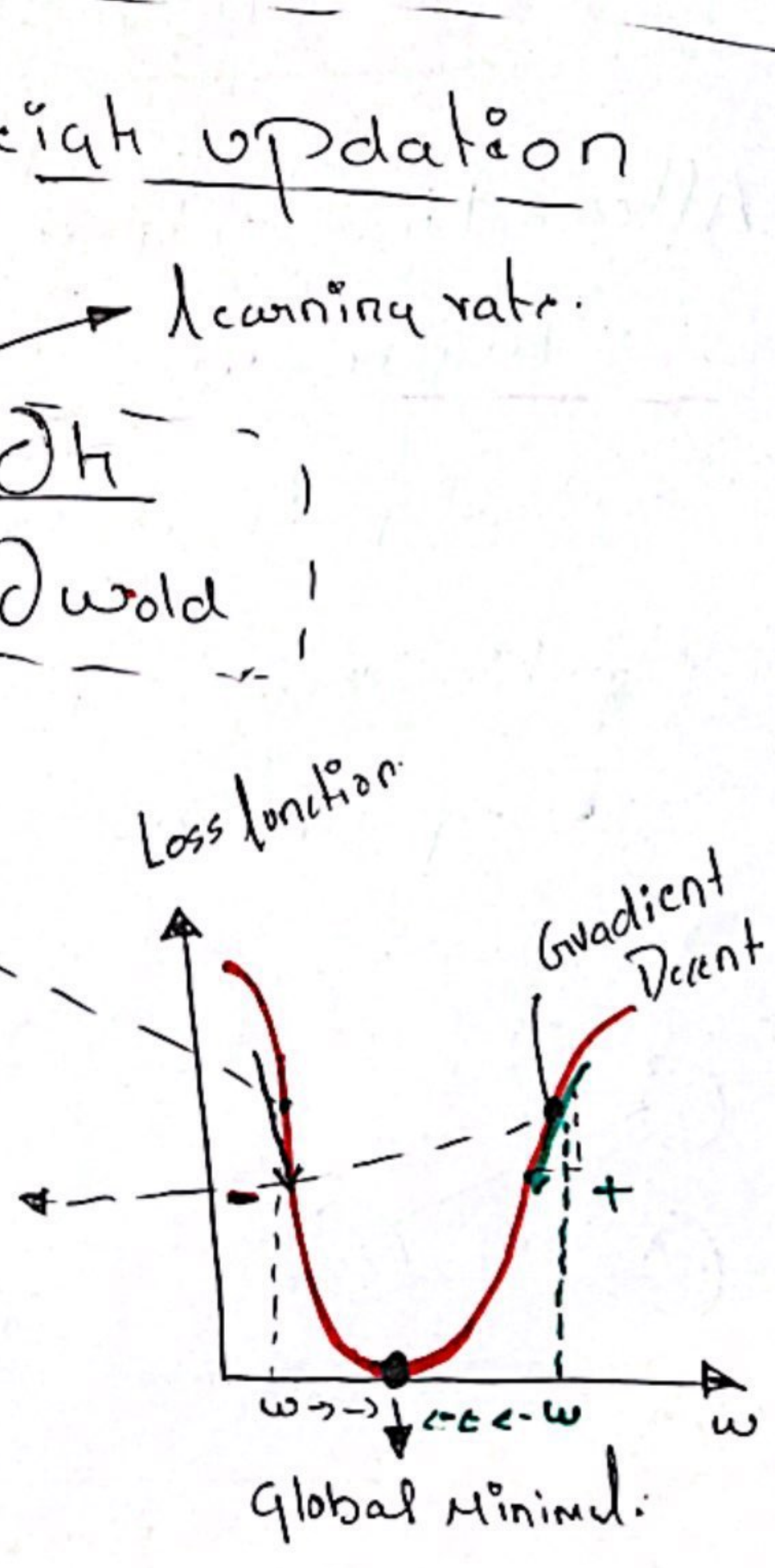
$$W_{new} = W_{old} - \eta(-v_e)$$

(-)

$$= W_{old} + \eta(+v_e)$$

$$W_{new} = W_{old} - \eta(+v_e)$$

$W_{new} < W_{old}$



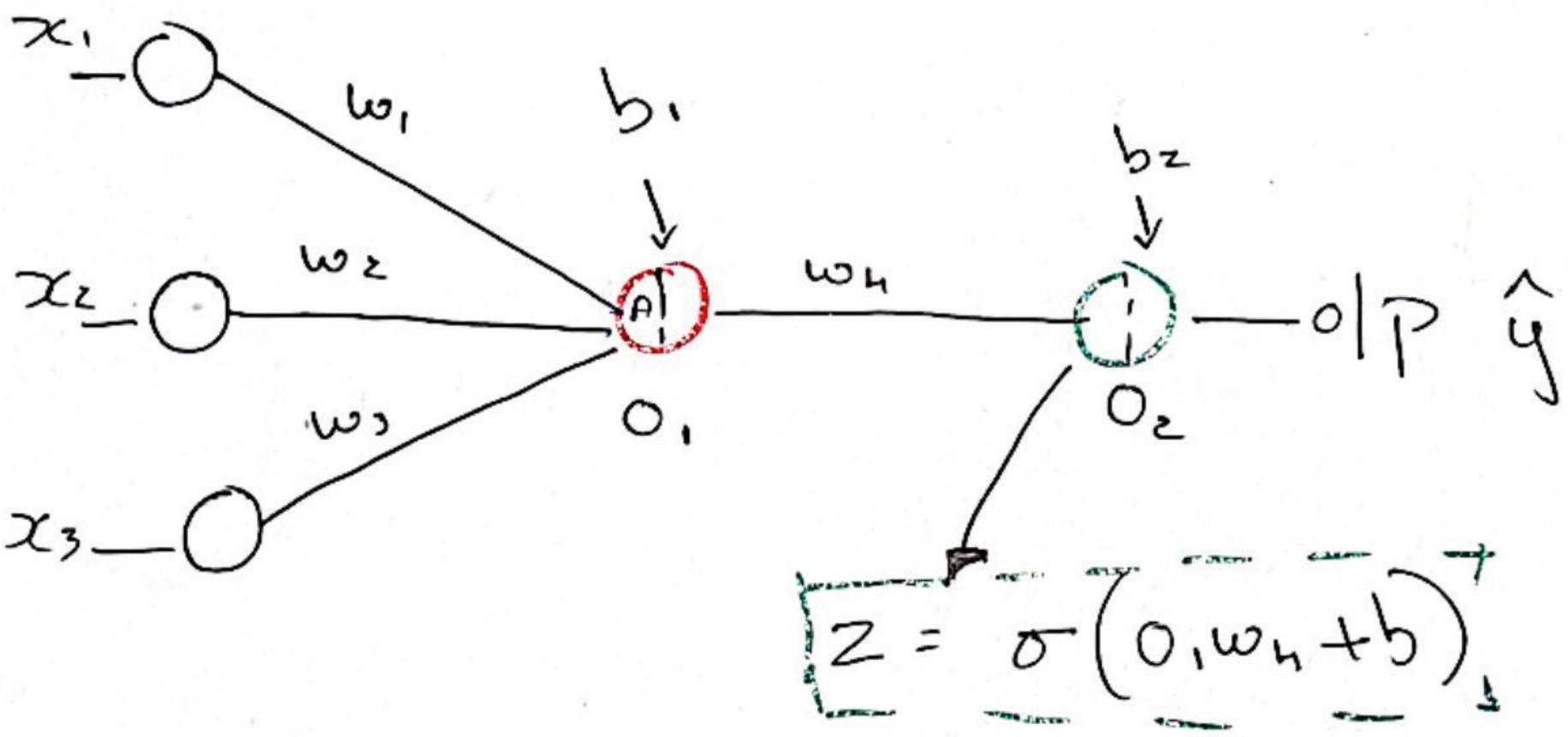
② Chain Rule of Differentiation

$$W_{new} = W_{old} - \eta \frac{\partial L}{\partial W_{old}}$$

- η → learning rate.
- $\frac{\partial L}{\partial W_{old}}$ → derivate of loss
- ∂W_{old} → derivd of W_{old}.

$(\sum w_i x_i) + b$

learning rate $\boxed{\eta = 0.01}$ or $\boxed{\eta = 0.001}$ best



$\eta = 0.001$

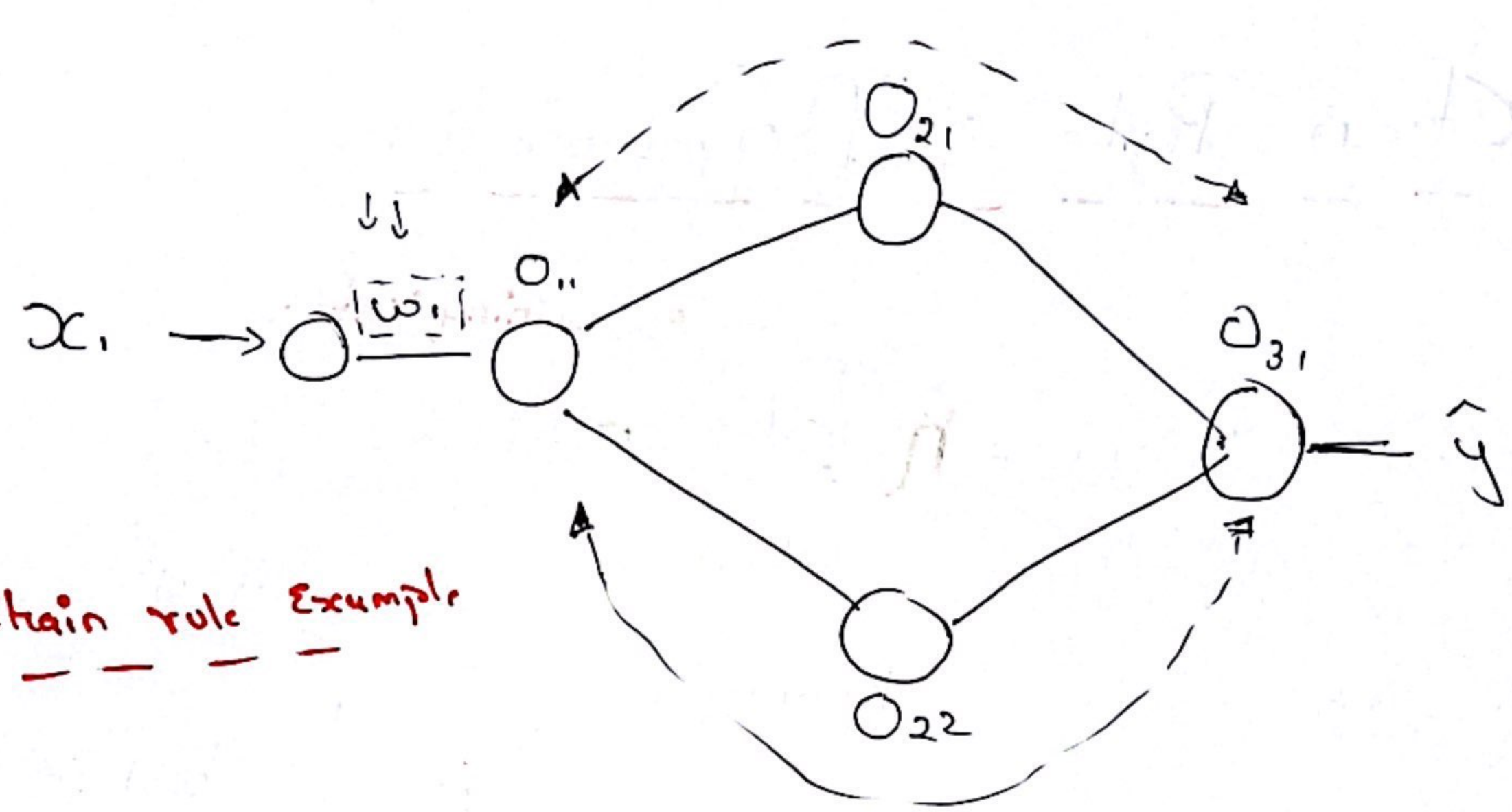
$$W_{4_{new}} = W_{4_{old}} - \eta \left[\frac{\partial L}{\partial w_{4old}}\right]$$

$o/p_2 = \sigma((w_4 * O_1) + b_2)$

$$\frac{\partial L}{\partial w_{4old}} = \frac{\partial L}{\partial O_2} * \frac{\partial O_2}{\partial w_{4old}}$$ → chain rule.

Bias update

$$b_{2_{new}} = b_{2_{old}} - \eta \frac{\partial L}{\partial b_{2old}}$$



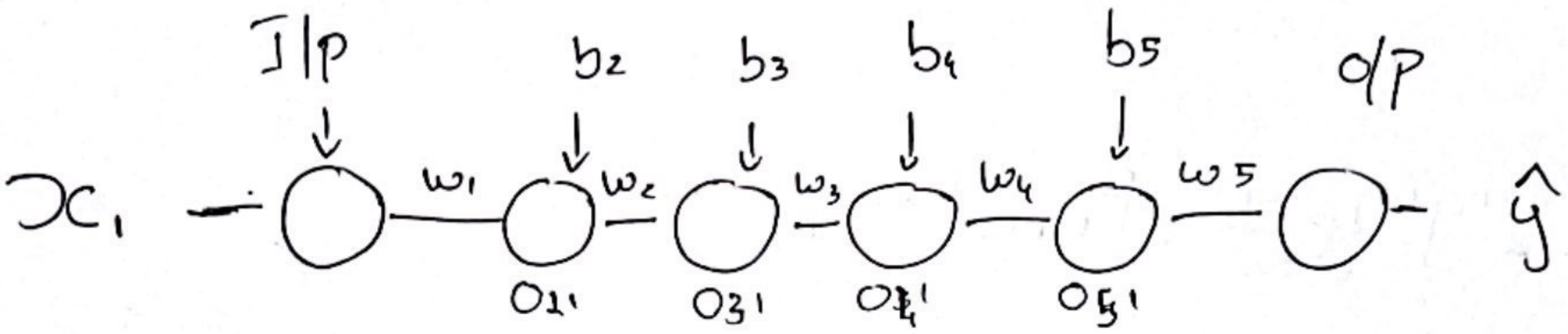
Chain rule Example

$$\omega_{1_{new}} = \omega_{1_{old}} - \eta \left(\frac{\partial L}{\partial \omega_{1,old}} \right).$$

$$\frac{\partial L}{\partial \omega_{1,old}} = \left[\frac{\partial L}{\partial O_{31}} \times \frac{\partial O_{31}}{\partial O_{21}} \times \frac{\partial O_{21}}{\partial O_{11}} \times \frac{\partial O_{11}}{\partial \omega_1} \right] +$$

$$\left[\frac{\partial L}{\partial O_{31}} \times \frac{\partial O_{31}}{\partial O_{22}} \times \frac{\partial O_{22}}{\partial O_{11}} \times \frac{\partial O_{11}}{\partial \omega_1} \right].$$

Vanishing Gradient Problem



MSE ⇒ $\boxed{\text{Loss} = \frac{1}{2}(y-\hat{y})^2}$

$$\omega_{1new} = \omega_{1old} - \eta \frac{\partial L}{\partial \omega_{1old}}.$$

To overcome VGD we Did AF

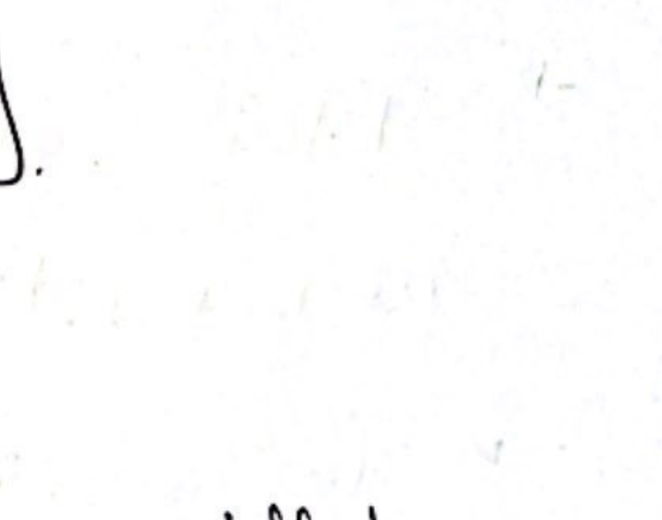
$$\frac{\partial L}{\partial \omega_{1old}} = \frac{\partial L}{\partial O_{51}} \times \frac{\partial O_{51}}{\partial O_{41}} \times \frac{\partial O_{41}}{\partial O_{31}} \times \frac{\partial O_{31}}{\partial O_{21}} \times \frac{\partial O_{21}}{\partial \omega_{1old}}.$$

$$O_{51} = \sigma\left[(O_{41} \times \omega_4) + b\right].$$

↳ sigmoid funct

σ derivation rule is value will be ranging in 0 to 0.25

$$\frac{\partial L}{\partial \omega_{1old}} = 0.25 \times 0.15 \times 0.10 \times 0.05.$$



when value are Hadly updating.

When $\omega_{1old} \overset{\text{Approx}}{\simeq} \omega_{1new}$ it is Called to be VGD. when the value are too closed. Or not updating.

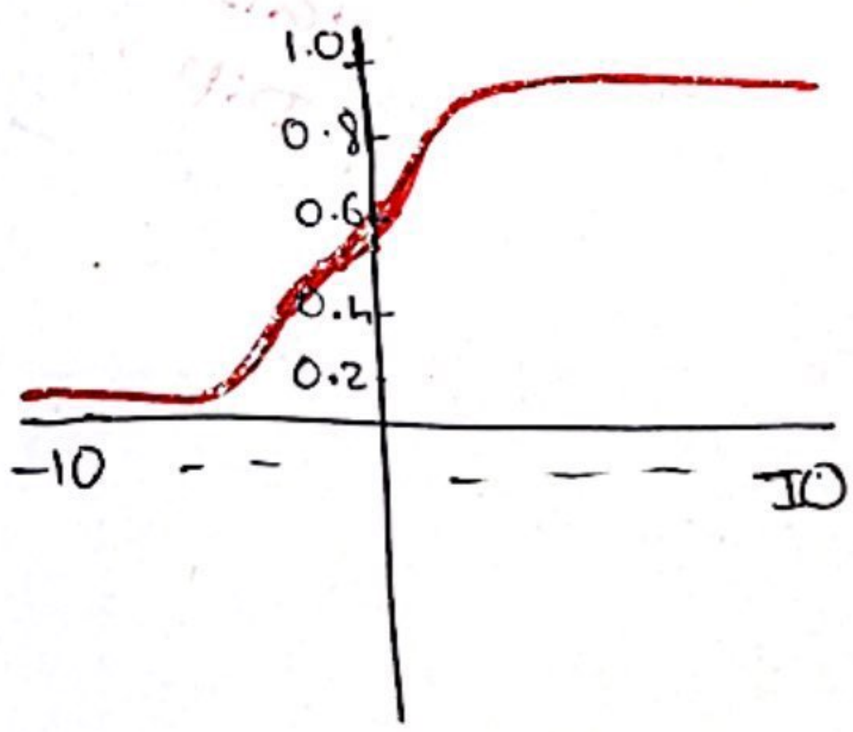
Activation Functions

① Sigmod Function

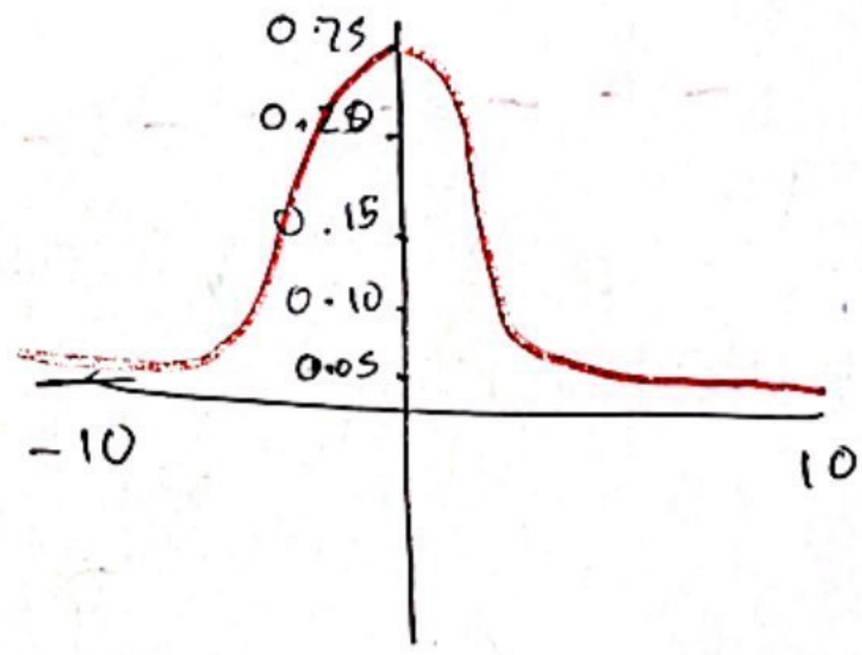
$$\sigma(x) = \frac{1}{1+e^{-x}}$$

Vanishing Gradient Problem.

$\sigma(x)$



$\partial \sigma(x)/\partial x$



→ It is used when we have to predict the probability as an output.

→ To add non-linearity in ML model

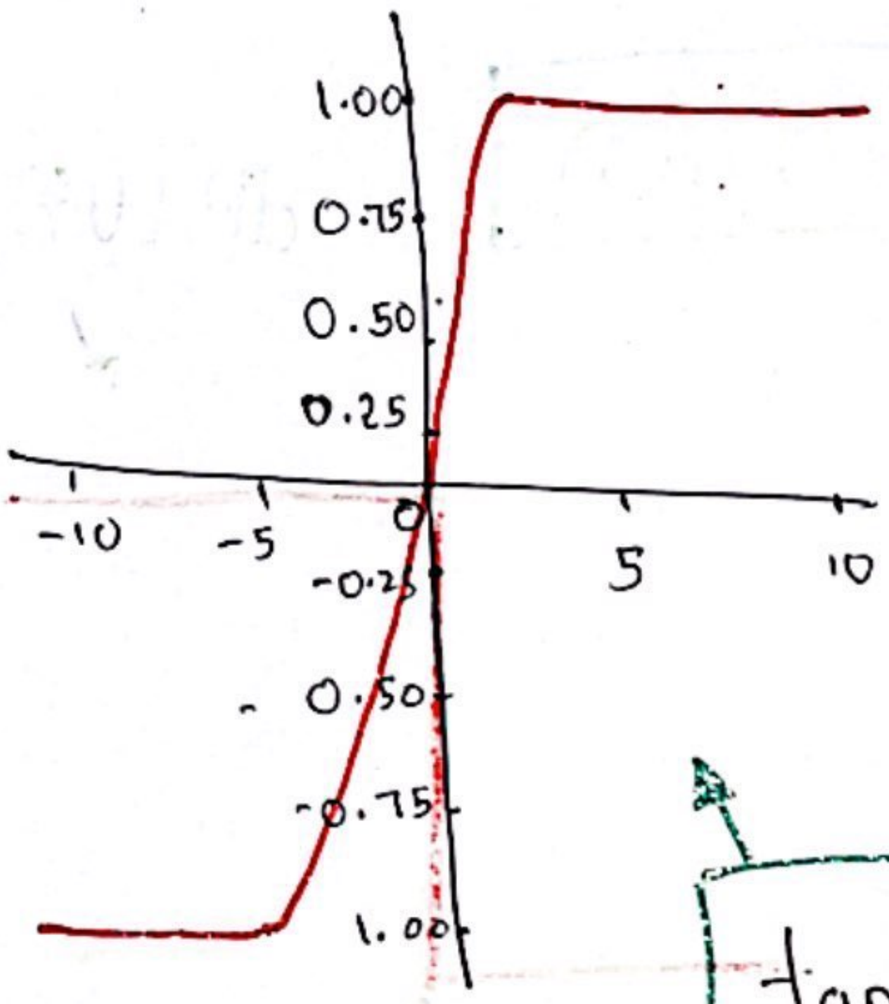
→ Mainly used for binary classification.

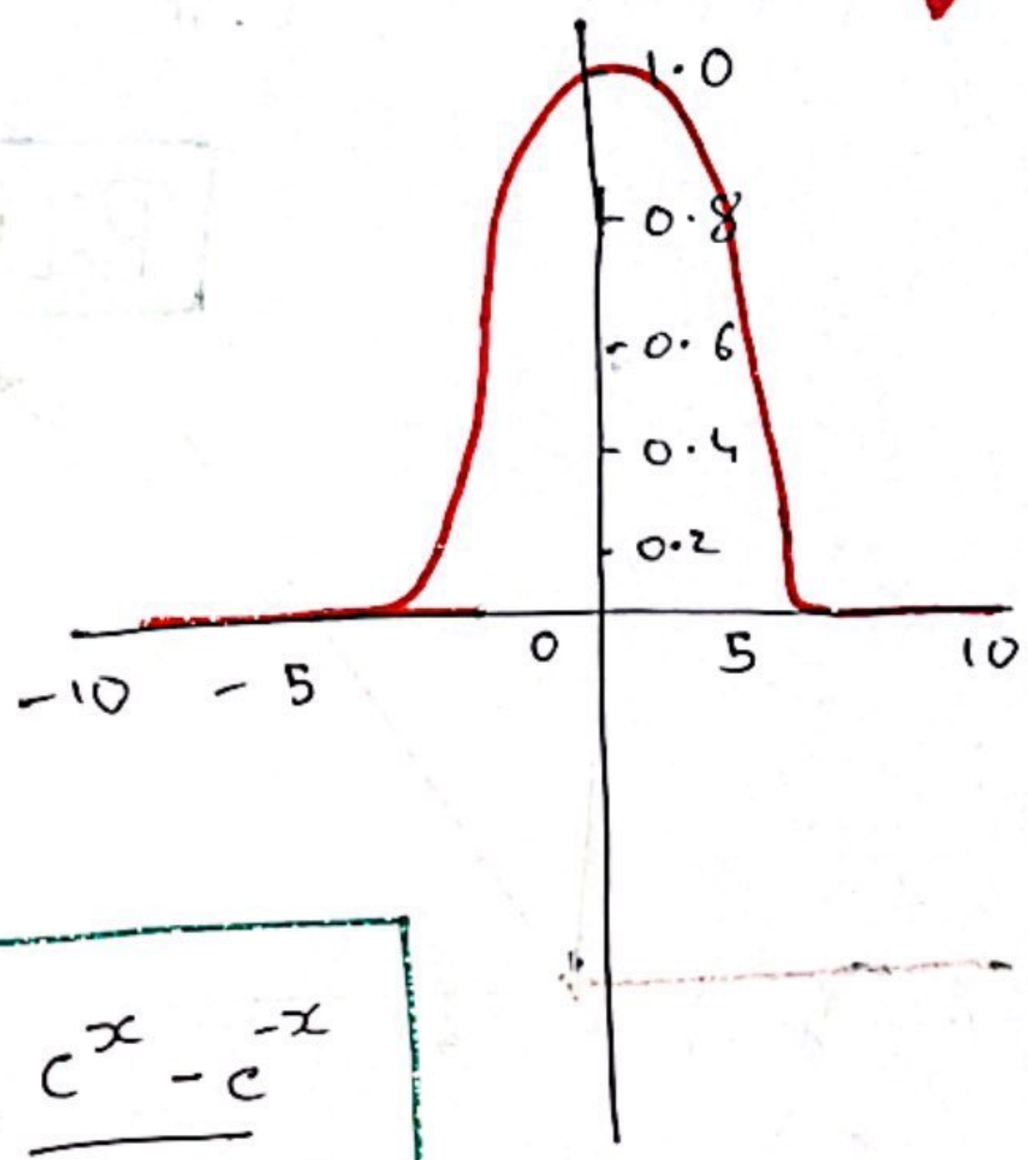
→ Advantage: Smooth gradient
Prevent jumps in output value.

→ Disadvantage: • Prone to gradient vanishing.
• Power operation are time consumed

(2) tanh Function

dtanh(x)/dx





$$\tanh x = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

→ It is a hyperbolic tanh curve.

→ Value (-1 to 1) derivate will be in vane (0 to 1)

It overcomes 0 centric Problem. (It is 0 centric).

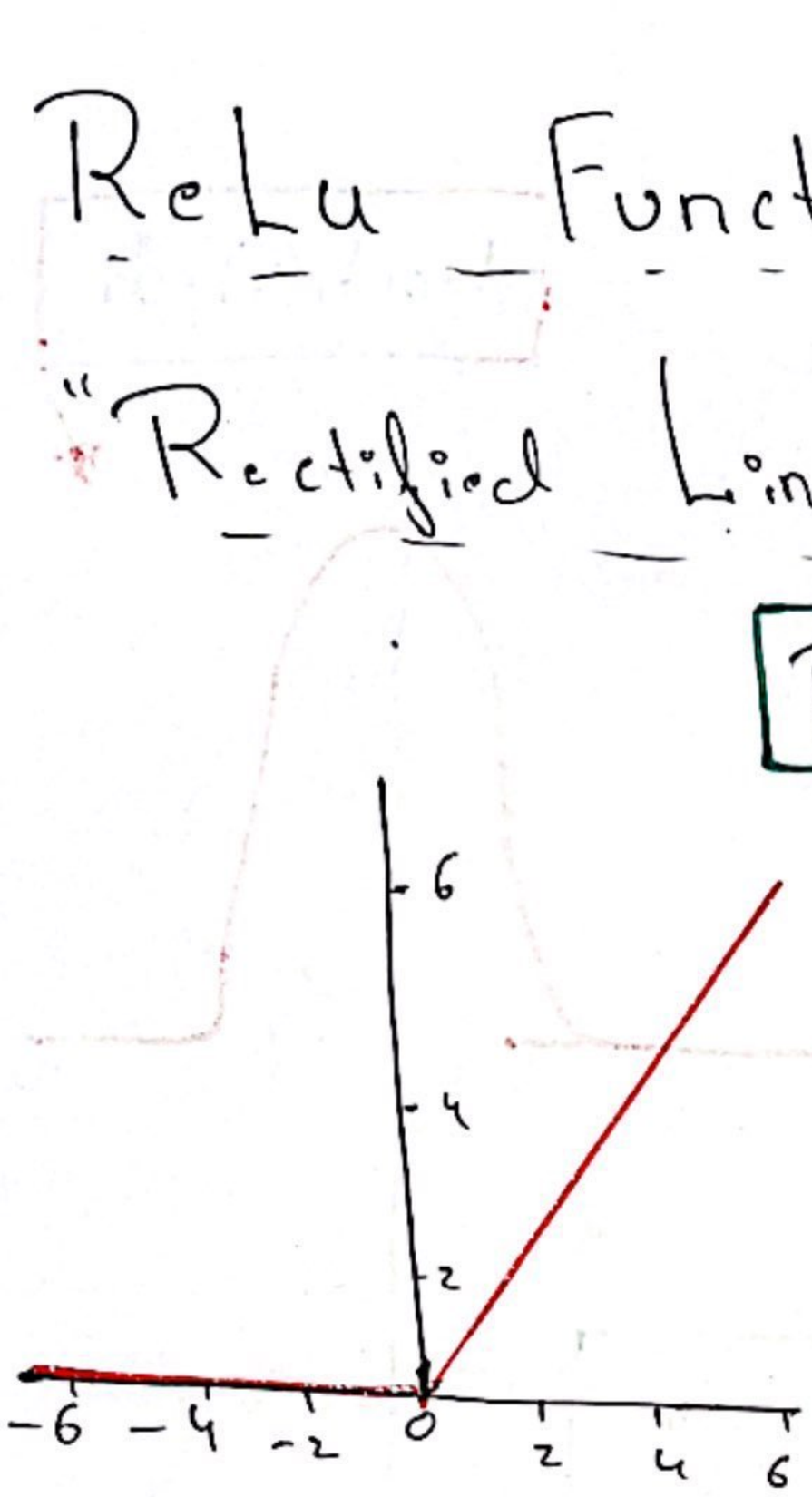
Note important :- in Binary classification problem the "tanh function is used for the Hidden layer" & "Sigmod function is used for the output."

→ There will be a Some small amt of VGD problem

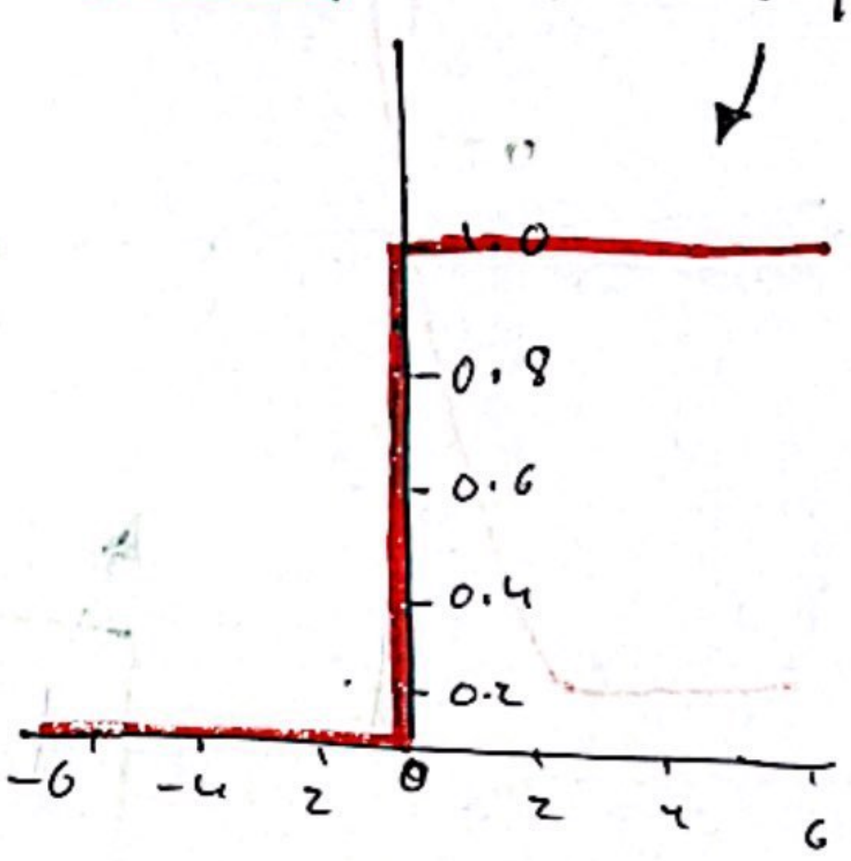
③ ReLu Function

"Rectified Linear Unite"

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



dReLU(x)/dx



→ When x value is -Ve ReLu will convert 0

◎ x value is -Ve x value will be same.

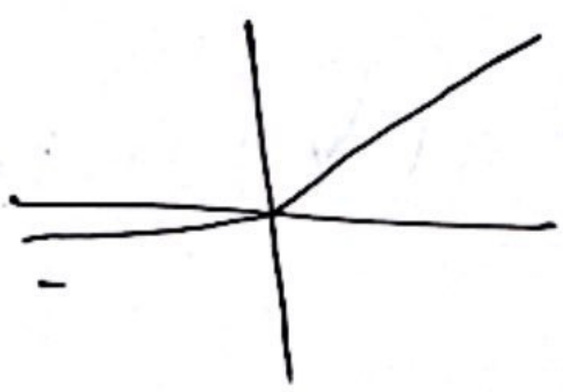
→ derivative will be 0 (or) 1

→ It Solving Sigmoid and Tanh function of VGD.

→ when input is -Ve ReLU will be inactive.
Neuron will die

④ Leaky Relu function

$$f(x) = max(0.01x, x)$$ (0.01 can change)

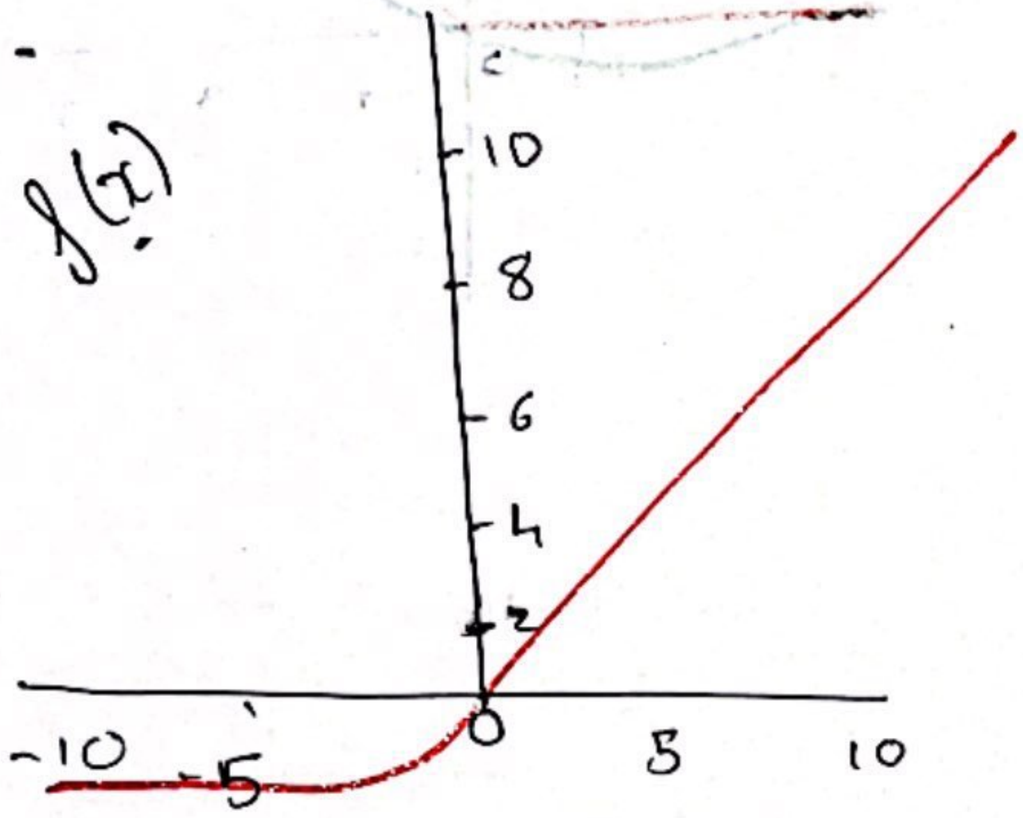
→ 

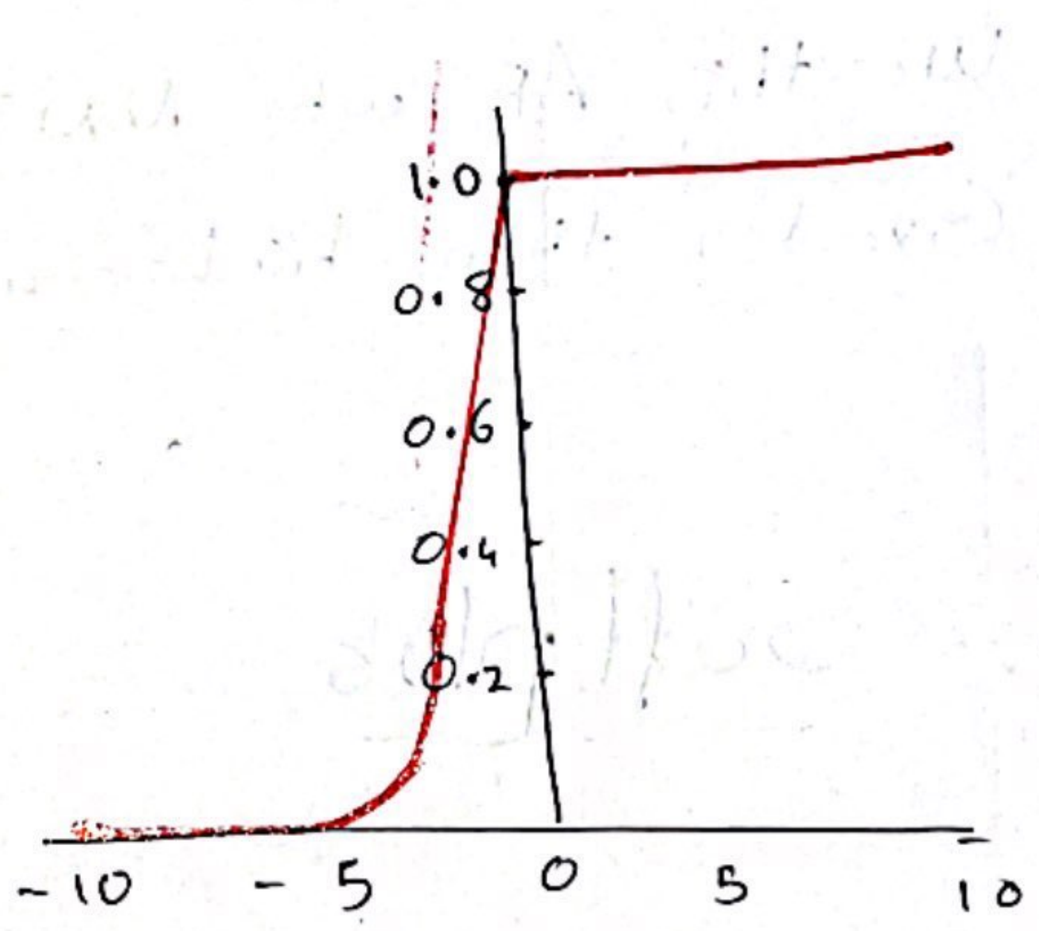
- -Ve value will be their in Small amt.
- It Solves die neron problem.

⑤ ELU (Exponential Linear Unit) function

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

$\partial f(x)/\partial x$





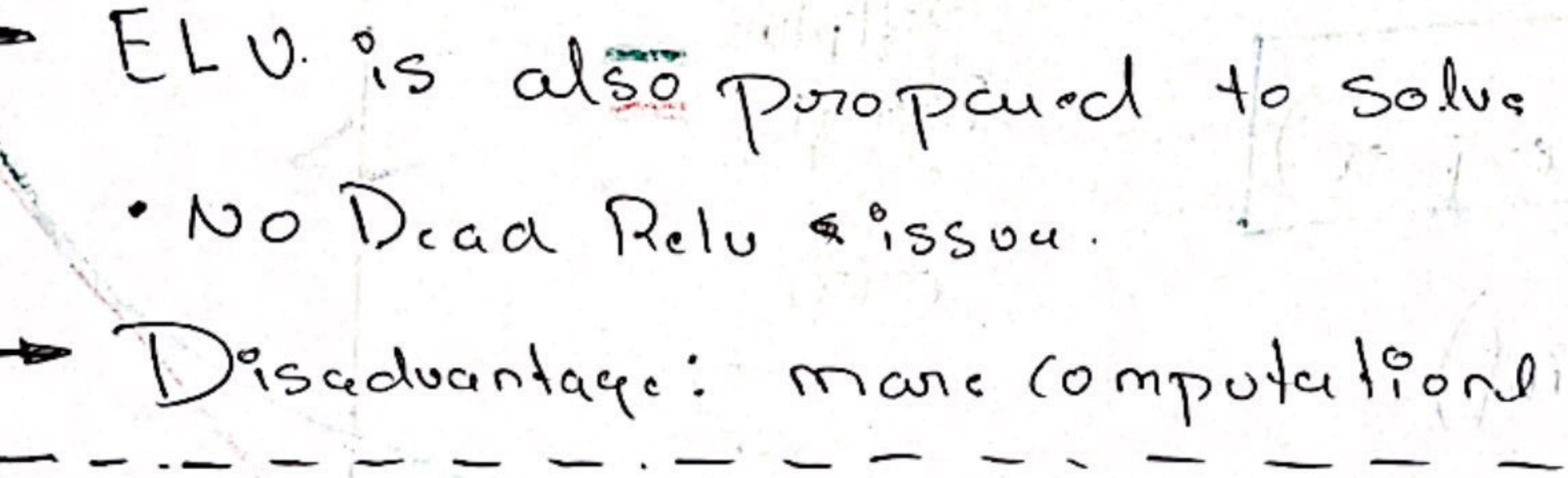
→ ELU is also propaed to solve ReLU problem.

- No Dead Relu issue.

→ Disadvantage: more computational intensiv.

---

⑥ P.ReLu (Parametric ReLu)



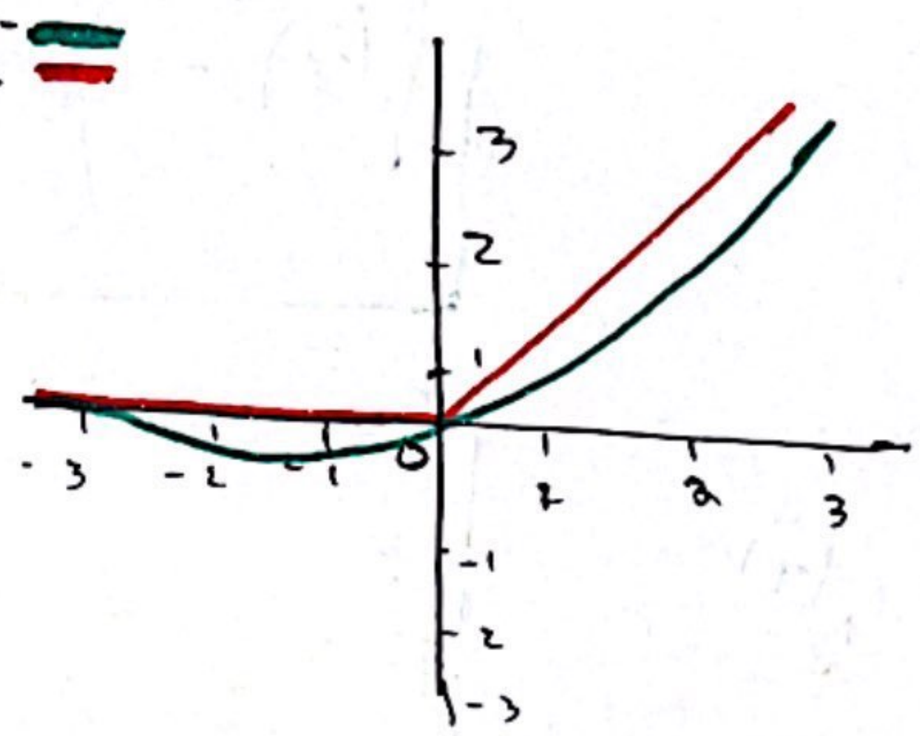
$$f(y_i) = \begin{cases} y_i & \text{if } y_i > 0 \\ a_i y_i & \text{if } y_i \leq 0 \end{cases}$$

- if $a_i = 0$, ∫ ReLu
- if $a_i > 1$, ∫ Leak ReLu
- if a_i is a ∫ RReLu

- Value will be in Range (0 & 1)

⑦ Swish (A self Gated) function. ReLu

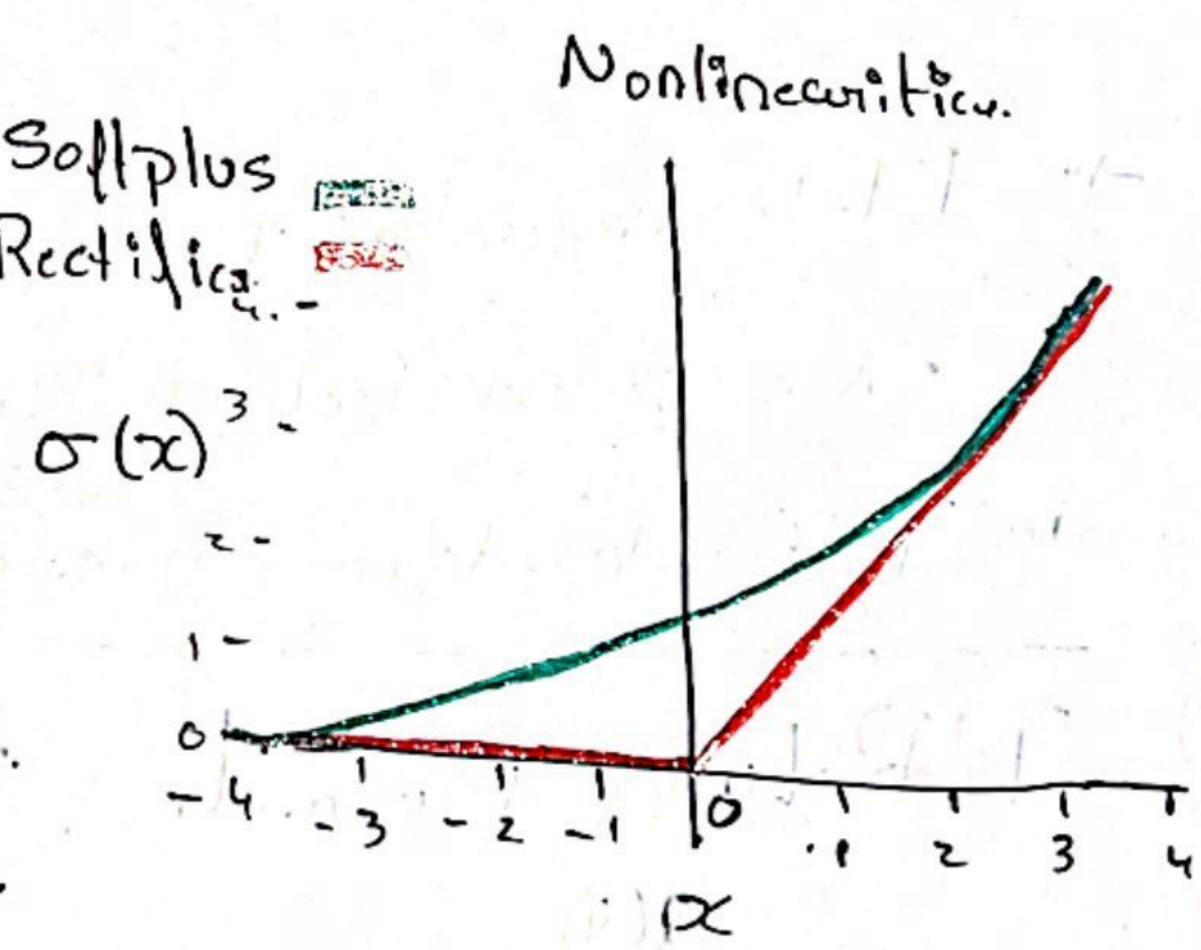
$$y = x * \text{Sigmoid}(x)$$



We this AF whe NN is Greater than 40 layers.

⑧ Softplus

$$f(x) = \ln(1 + \exp x)$$



- ⊙ Range: (0, +inf)
- ⊙ Similar & relatively Smooth.
- ⊙ It is unilateral suspension.

Relu

⑨ Maxout

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

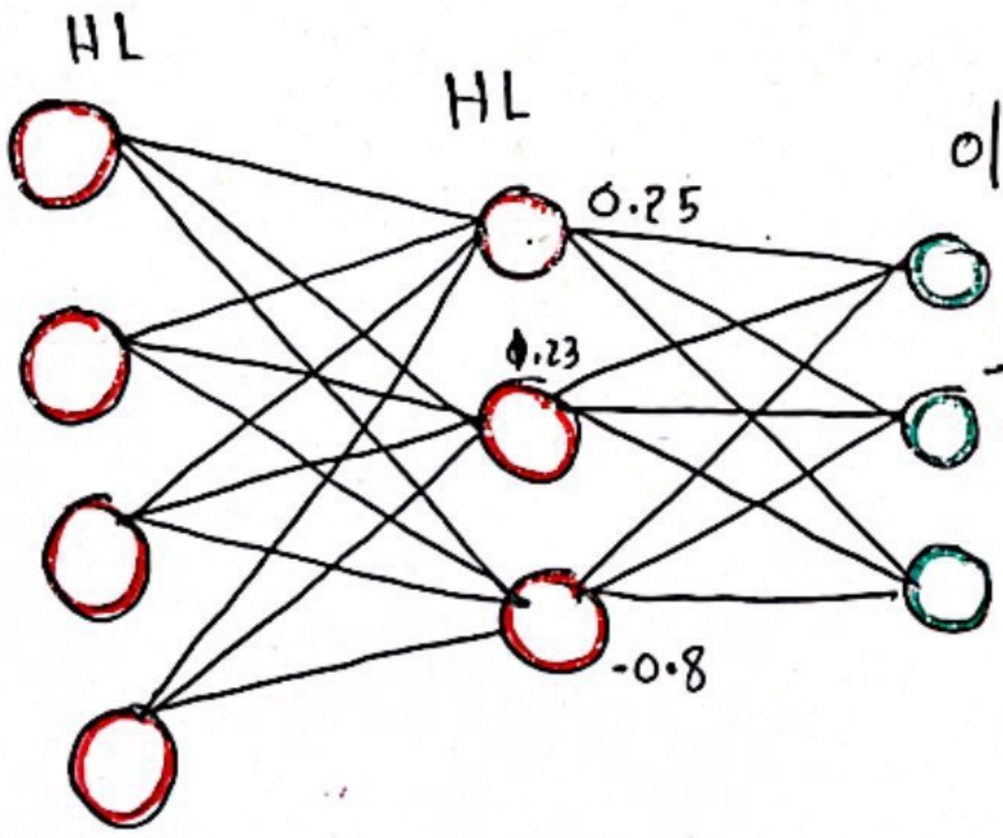
- ⊙ Parameter K.
- ⊙ ReLu & Leak Relu is Generalised Here.
- ⊙ Output will be largest activation value.

(10) # Softmax

$$S(x_j) = \frac{e^{x_j}}{\sum_{k=1}^{k} e^{x_k}} \quad , 1, 2, \ldots, k.$$

- Softmax is used when output > 2 layer.
- Output will be in the form of probality. Highest Probality will be considered.
- It is always applied at "output layer".

Example: identity.



o/p $P_r[x_j = \text{Seal}] = \text{Softmax}(z)_0$.

$\frac{e^{0.25}}{e^{0.25} + e^{1.25} + e^{-0.8}} = 0.249.$

$\frac{e^{1.25}}{e^{0.25} + e^{1.25} + e^{-0.8}} = 0.664$ x_j = Panda.

$\frac{e^{-0.8}}{e^{0.25} + e^{1.25} + e^{-0.8}} = 0.087$ x_j = dock.

Note :- Sigmoid and Softmax are always used for Output layer if ≤ 2 out layer Sigmoid
if > 2 out layer Softmax.

Notes

① ReLu is Most Prefferen for hidden layer.

② → Binary clarification output layer use Sigmoid.
→ Multi clarification output layer use Softmax.
clan

③ for regression like problem. to active continous value use linear activation function in Output layer.

④ Tanh is use in Hidden layer in Binary. Classification.

⑤ Swish is used only when there is more than 40 Layers in NN.

⑥

Evaluation Methods.

① Regression.

Loss function: $\frac{1}{2}(y-\hat{y})^2$ ← Single Record

cost function: $\frac{1}{2}\sum_{i=1}^{n}(y-\hat{y})^2$ ← Batch of Records

ⓐ Mean Square Error (MSE)

Loss function = $\frac{1}{2n}(y-\hat{y})^2$

cost function = $\frac{1}{2n}\sum_{i=1}^{n}(y-\hat{y})^2$

Advantage ① Differentiable.

② It has only one local or Global Minima.

③ It convergent faster.

disadvantage: Robust to out layers.

ⓑ Mean absulate Error (MAE).

Los function = $\frac{1}{n}|y-\hat{y}|$

cost function = $\frac{1}{n}\sum_{i=1}^{n}|y-\hat{y}|$

Advantage: over come the Robust to outlayers.

- No Square is done.

disadvantage: Time consume.

⑥ Huber loss

combination of both MSE & MAE.

$$\text{Loss} = \begin{cases} \frac{1}{2}(y-\hat{y})^2 & \text{if } |y-\hat{y}| \le \delta \\ \delta|y-\hat{y}| - \frac{1}{2}\cdot\delta^2 & \text{otherwise. No outliers.} \end{cases}$$

Hyperparameter

δ → outliers

② Classification

Cross Entropy →
- Binary Cross Entropy. (Binary classification)
- Categorical CRoss Entropy. (Multiclass classification).

ⓐ Binary Cross Entropy.

$$\text{Loss} = -y * \log(\hat{y}) - (1-y) * \log(1-\hat{y}) = \text{LR}$$

$$\text{loss} = \begin{cases} -\log(1-\hat{y}) & \text{if } y=0 \\ -\log(\hat{y}) & \text{if } y=1 \end{cases}$$

$\hat{y}$ = sigmoid function. $\frac{1}{1+e^{-x}}$

(b) Cataqorical Cross Entropy.

$$loss(x_i, y_i) = -\sum_{j=1}^{c} y_{ij} * \ln(\hat{y}_{ij})$$

$y_i = [y_{i1}, y_{i2}, y_{i3}, ---- y_{ic}]$.

$y_{ij} = \begin{cases} 1 & \text{if the element is in class.} \\ 0 & \text{otherwise.} \end{cases}$

$\hat{y}_{ij}$ = Soft Max Activation function.

Deep learning Optimizers terms

① **Epoch** - The number of times algorithm runs on the whole training dataset.

② **Sample** - A Single row of dataset

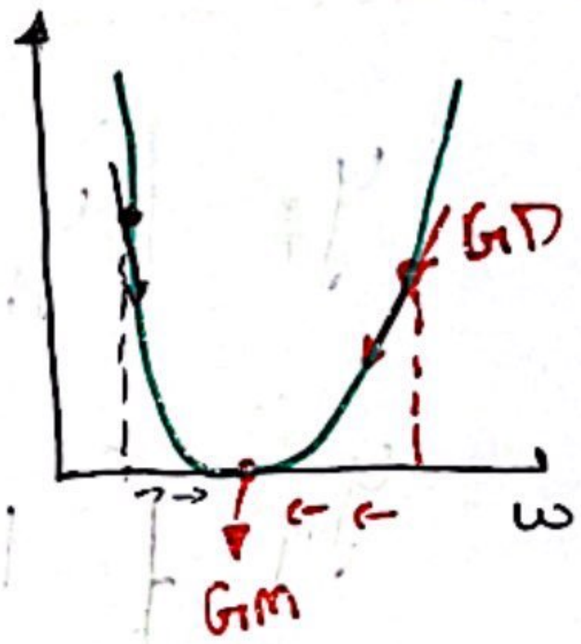
③ **Batch** - the number of Samples to be taken for updating model parameter.

④ **learning rate**:- It is a Parameter. that provided. the model a Scale of how much ~~model~~ parameter. weight Should be update.

Optimization

① Gradient Descent

loss/cost



weight updation formula.

$$w_{new} = w_{old} - \eta \left[\frac{\partial h}{\partial w_{old}} \right]$$

learning rate

Chain rule derivative.

$$\left(\sum w_i x_i\right) + b$$

- Disadvantage
  - Resource Expensive.

* Do not use this if dataset is huge.

② Stochastic Gradient Descent

- To overcome GD compolation cost ↑ GD was derived.
- Stochastic means randomness. We randomly Select the Samples and make that as Batch.

"Batch size = No of random Samples."

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

- SGD takes more Iterations to reach local minima.

⊙ It is increase training time but decreases computation cost. fed Single record to algorithm

Disadvantage

- increase in training time.
- increase in noise cuz of randomness.

③ Mini Batch SGD

- Batch size is a parameter to make group of random sample.
- How to choose perfect batch size?

Ans:-

Batch size = 10,000 ← Dataset = 1,00,000

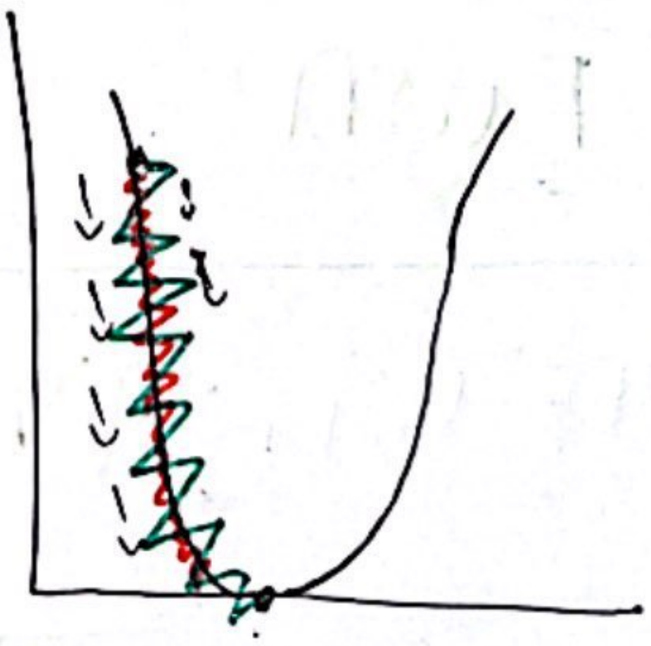
Iteration = 10 randomely selected.

Once cycle is 1 EPOCH.

- Convergence is better
- time will improve.
- comparatively less resource expensive.
- Reduces the noise

Disadvantage

- increasing in train time.
- Noise in data.



④ SGD with Momentum

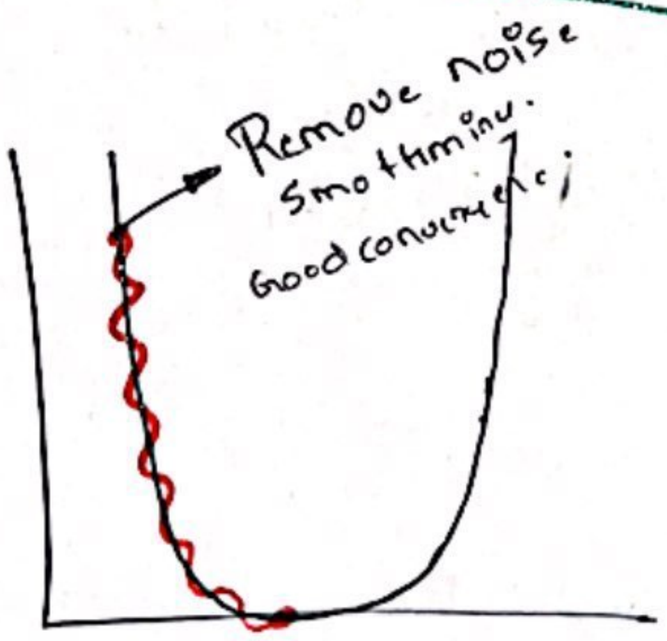
To over come the problem of noise we use SGD with Momentum.

$$w_{new} = w_{old} - \eta \frac{\partial L}{\partial w_{old}}.$$

$$b_{new} = b_{old} - \eta \left[\frac{\partial L}{\partial b_{old}} \right]$$

We use "Exponential weighted average"
used in time Series.

$$w_t = w_{t-1} - \eta \frac{\partial L}{\partial w_{t-1}}$$



$t_1 \quad t_2 \quad t_3 \quad t_4 \ldots t_n$

$a_1 \quad a_2 \quad a_3 \quad a_4 \ldots a_n$

$\beta = 0$ to 1
$= 0.95$

$$V_{t_1} = a_1$$

$$V_{t_2} = \beta * V_{t_1} + (1-\beta) * a_2.$$

$$= (0.95) * V_{t_1} + (0.05) * a_2$$

$$V_{t_3} = \beta * V_{t_2} + (1-\beta) * a_3.$$

EWA

$$w_t = w_{t-1} - \eta [V_{dw}]$$

$$V_{\partial w} = \beta * V_{\partial w_{t-1}} + (1-\beta) * \frac{\partial L}{\partial w_{t-1}}$$

weight updating formula

⑤ Adagrad (Adaptive Gradient Descent)

It Majorly concentrate on learning rate η

⊙ Learning rate is constent. in this Adagrad. h will be adaptive.

$$w_t = w_{t-1} - \eta' \frac{\partial L}{\partial w_{t-1}}$$

$$\eta' = \frac{\eta}{\sqrt{\partial h_t + (\epsilon)}}$$

Small num not to make zero.

$$h_t = \sum_{i=1}^{t} \left(\frac{\partial L_i}{\partial w_t}\right)^2 \uparrow\uparrow\uparrow$$

• Note : η Sohould be decreced when it reachem near to Global minma.

⊙ LR will be decreased when it is near to global minima.

Disadvantage

→ In deep neral network. when η will decreases. it might Effect

• decreing fast so hard to find GM

⊙ Advantage.

⊙ It reachem convergence at a higher speed.

---

⑥ Adadelta. and RMSProp

Exponential weighted Averages.

$$SDw = 0$$

$$SDw_t = \beta\, SDw_{t-1} + (1-\beta)\left(\frac{\partial L}{\partial w_{t-1}}\right)^2 \uparrow\uparrow\uparrow$$

⊙ Main aim is to control increase in $\left(\frac{\partial L}{\partial w_{t-1}}\right)^{\varepsilon}$ ↑↑↑ slow

so η will be controlled.

⑦ Adam Optimizer ★★★

Adaptive moment estimation.

It is the combination of Momentum + RMSprop.

Both Adaptive learning rate and Smoothing is also done together.

⊙ Smoothing

⊙ Learning is Adaptive.

Procs

$$w_t = w_{t-1} - \eta' V_{dw} \rightarrow (1a) \qquad \eta' = \frac{\eta}{\sqrt{S_{dw_t} + \epsilon}} \rightarrow (2)$$

$$b_t = w_{bt-1} - \eta' V_{db} \rightarrow (1b)$$

$$V_{dw_t} = \beta * V_{dw_{t-1}} + (1-\beta)\frac{\partial L}{\partial w_{t-1}} \rightarrow (3)$$

$$V_{db_t} = \beta * V_{db_{t-1}} + (1-\beta)\frac{\partial L}{\partial b_{t-1}} \rightarrow (4)$$

where S_{dw} is

$$S_{dw_t} = \beta S_{dw_{t-1}} + (1-\beta)\left(\frac{\partial L}{\partial w_{t-1}}\right)^2 \rightarrow 5$$

(5) in (2)

(2), (3) & (4) in 1a & 1b

Deep learning Libraries

① "from tensorflow.keras.models import Sequential"

Library. Library. Holds DL model.

Sequential — ANN is called/initialize using Sequential.

Wholeke set of input will be taken in the form of blocks. It can perform all the forward and Backward propogation in Sequencial way.

② "from tensorflow.keras.layers import Dense."

Dill layers.

Dense

Creating and managing the neurons. And for modifiyin the hidden layers neurons.

- Create hidden layers
- Input and Output layers creation.

③ "from tensorflow.keras.layers import ReLU, LeakyRelu, PRelu, ELU, ,,"

modules.

imparing Activation functions. for neurons.

(4) "from tensorflow.keras.layers import Dropout."

- To overcome the Overfitting problem in Dataset we use Dropout.
  - used to solve complex Problem.
  - Similar to regularization in ML.

When we specify the Parameter of dropout. Assigned
ex: if 0.3 is specified 30% of the neurons gets deactivated.

(5) import tensorflow.

opt = tensorflow.keras.optimizers.Adam(learning_rate=0.01)

↳ specify in .compile(.)

Karthik Mahadiy.