

# 5 Multi layered Perceptron Model [Artificial Neural N/w]

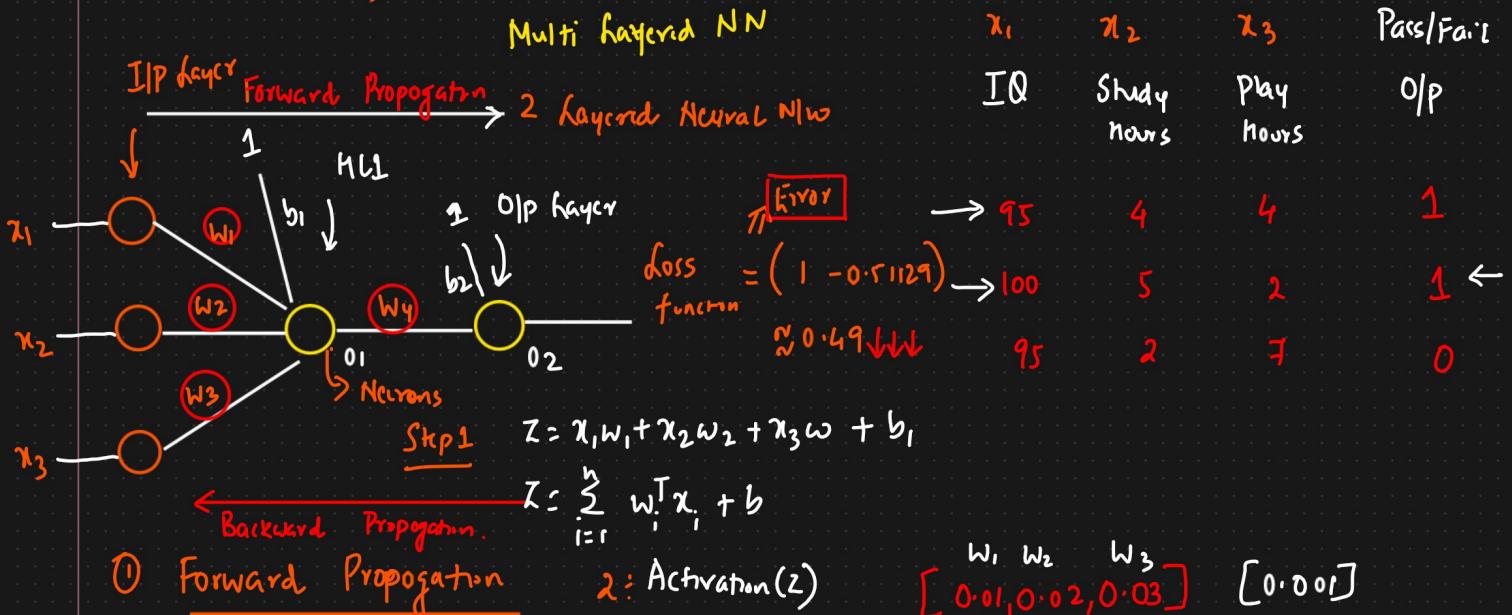
① Forward Propogation

② Backward Propogation → Geoffrey Hinton →

③ Loss function

④ Optimizers ✓ Gradient Descent.

⑤ Activation function.

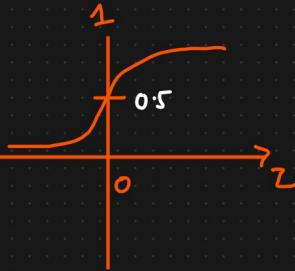


## Hidden Layer 1

$$\begin{aligned} \text{Step 1: } Z &= 95 \times 0.01 + 4 \times 0.02 + 4 \times 0.03 + 1 \times 0.01 \\ &= 1.151 \end{aligned}$$

Sigmoid.

$$= \frac{1}{1+e^{-Z}}$$



Step 2: Activation (z)

$$f(z) = \frac{1}{1+e^{-1.151}} = 0.759$$

$$0_1 = 0.759$$

## Hidden Layer 2

$$w_4 = 0.02$$

$$b_2 = 0.03$$

$$\text{Step 1: } Z = 0_1 * w_4 + b_2$$

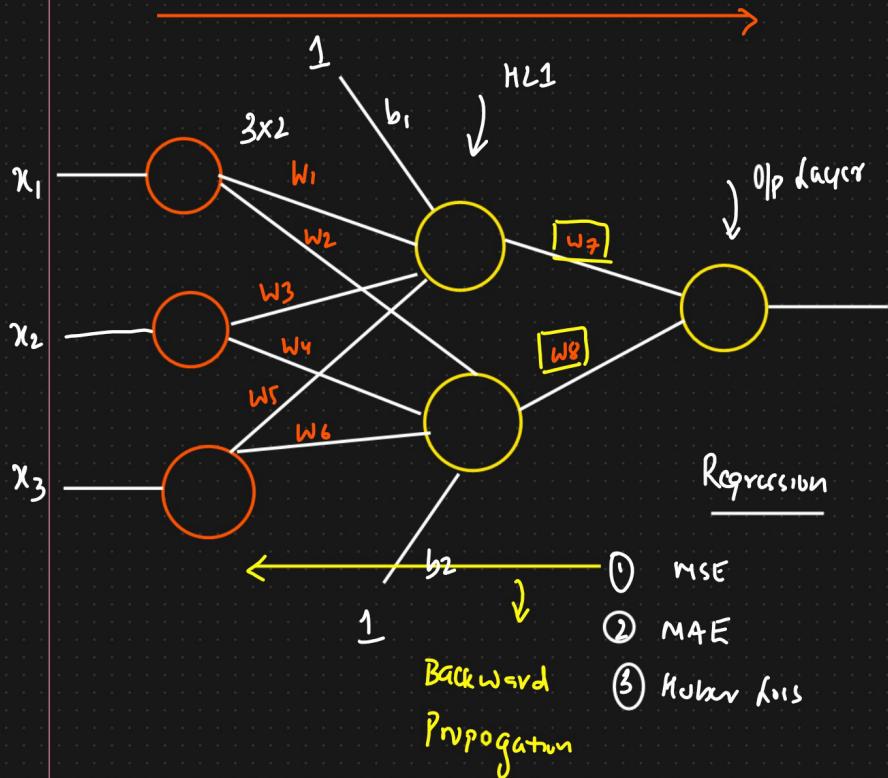
$$= 0.759 * 0.02 + 0.03$$

$$= 0.04518$$

Step 2 : Activation(z)  $\frac{1}{1+e^{-(0.04518)}} = 0.51129$

$$O_2 = 0.51129 \Rightarrow \hat{y}$$

## ⑥ Back Propagation And Weight Updation Formula



### Weight update Formula

$$w_{7\text{new}} = w_{7\text{old}} - \eta \left[ \frac{\partial L}{\partial w_{7\text{old}}} \right]$$

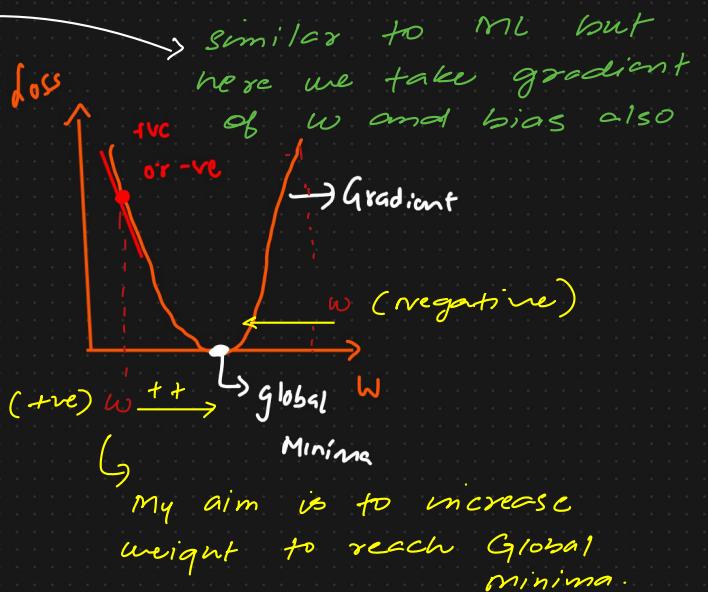
slope

$$w_{8\text{new}} = w_{8\text{old}} - \eta \left[ \frac{\partial L}{\partial w_{8\text{old}}} \right]$$

learning Rate

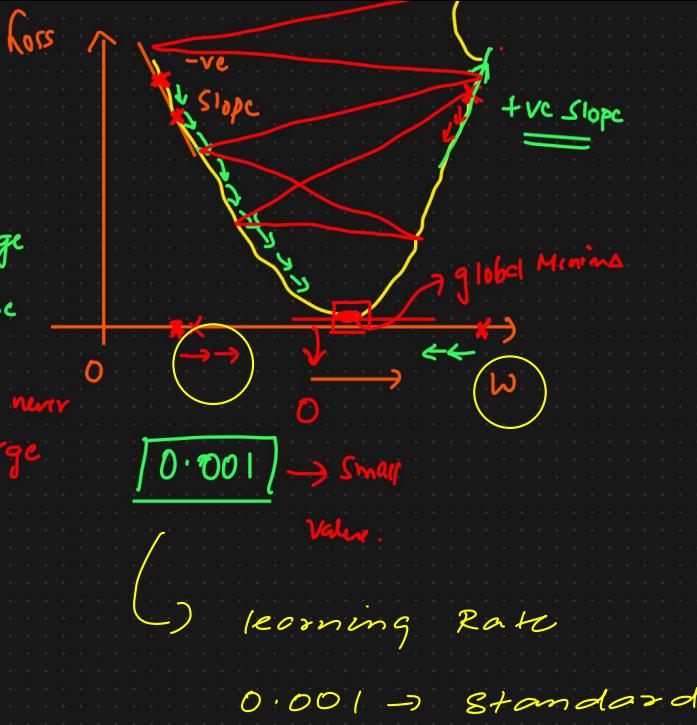
V.V. Imp

$$w_{\text{new}} = w_{\text{old}} - \eta \left[ \frac{\partial L}{\partial w_{\text{old}}} \right] \Rightarrow \text{Weight Updation Formula.}$$



Gradient Descent  
Optimizers

Optimizers : To reduce the loss value



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

$$= w_{\text{old}} + \eta (\text{+ve})$$

$$\boxed{w_{\text{new}} > w_{\text{old}}}$$

$\eta$  = large value

It may never converge

Learning Rate

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

$$= w_{\text{old}} - \eta (\text{+ve})$$

$$\boxed{w_{\text{new}} < w_{\text{old}}}$$

Learning Rate

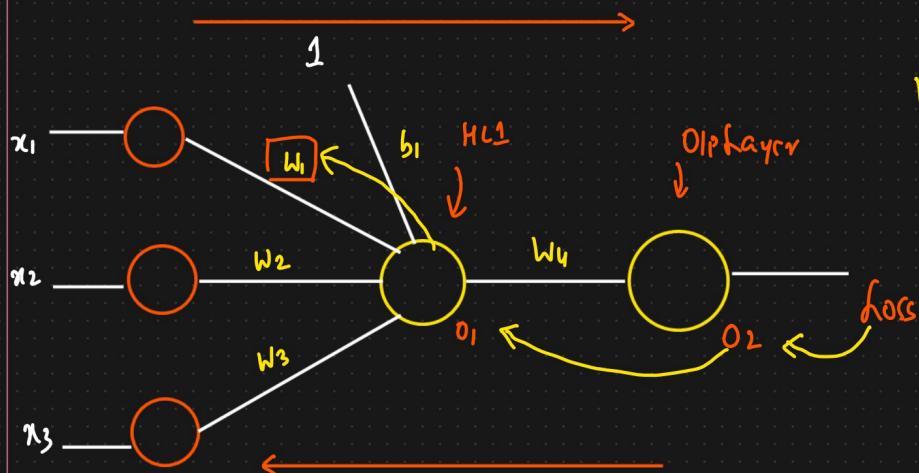
0.001 → Standard.

When  $w$  reaches Global Minima

$$\boxed{w_{\text{new}} = w_{\text{old}}}$$

→ used to find derivatives of composite functions (function inside a function).

### ⑦ Chain Rule of Derivative



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

$$w_{4_{\text{new}}} = w_{4_{\text{old}}} - \eta \boxed{\frac{\partial h}{\partial w_{4_{\text{old}}}}}$$

$$\frac{\partial h}{\partial w_{1,old}} = \frac{\partial h}{\partial o_2} * \frac{\partial o_2}{\partial w_{1,old}} \Rightarrow \text{Chain Rule of Derivation}$$

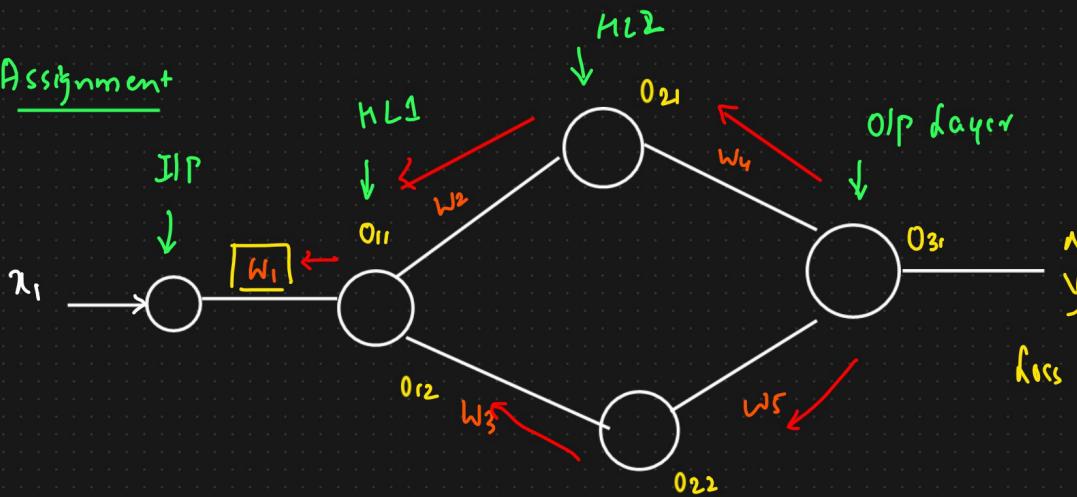
$$w_{1,new} = w_{1,old} - \eta \left[ \frac{\partial h}{\partial w_{1,old}} \right]$$

$$\frac{\partial h}{\partial w_{1,old}} = \frac{\partial h}{\partial o_2} * \frac{\partial o_2}{\partial o_1} * \frac{\partial o_1}{\partial w_{1,old}}$$

$w_{2,new}$

$w_{3,new}$

Assignment



$$w_{1,new} = w_{1,old} - \eta \left[ \frac{\partial h}{\partial w_{1,old}} \right]$$

↓

$$\frac{\partial h}{\partial w_{1,old}} = \left[ \frac{\partial h}{\partial o_{31}} * \frac{\partial o_{31}}{\partial o_{21}} * \frac{\partial o_{21}}{\partial o_{11}} * \frac{\partial o_{11}}{\partial w_{1,old}} \right]$$

+

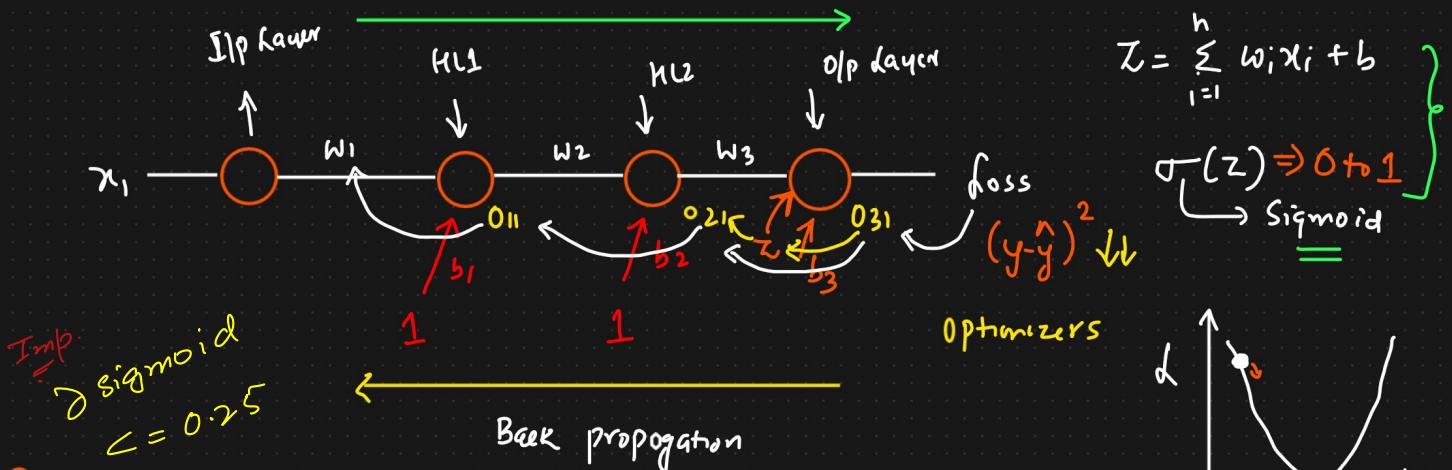
$$\left[ \frac{\partial h}{\partial o_{31}} * \frac{\partial o_{31}}{\partial o_{22}} * \frac{\partial o_{22}}{\partial o_{12}} * \frac{\partial o_{12}}{\partial w_{1,old}} \right] \quad \begin{matrix} \partial h \rightarrow \partial w_{1,old} \\ (\text{Chain Rule}) \end{matrix}$$

vv imp  
==

Backpropagation = Chain Rule Repeated Backward

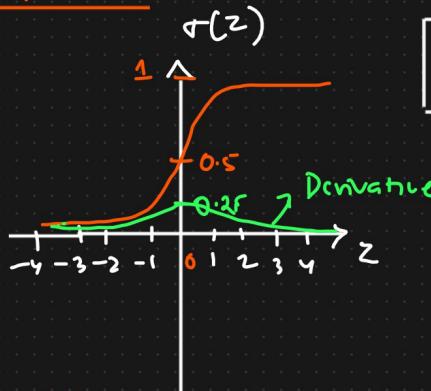
- Output Layer: Compute how loss changes with output (easy)
- Then apply chain rule to pass gradient back through each layer
- Eventually, you get the gradient of loss with respect to every weight

## (Solution = Change Activation Function) ⑧ Vanishing Gradient Problem And Activation functions



### ⑨ Sigmoid Activation function

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



$$0 \leq \sigma(z) \leq 1$$

Vanishing  
Gradient  
Problem.

$$\text{Derivative } (\sigma(z))$$

$$0 \leq \sigma'(z) \leq 0.25$$

$$w_{1,\text{new}} = w_{1,\text{old}} - \eta \left[ \frac{\partial h}{\partial w_{1,\text{old}}} \right] \Rightarrow \text{small value} \Rightarrow \boxed{w_{1,\text{new}} \approx w_{1,\text{old}}}$$

$$\frac{\partial h}{\partial w_{1,\text{old}}} = \frac{\partial h}{\partial o_{31}} * \frac{\partial o_{31}}{\partial o_{21}} * \frac{\partial o_{21}}{\partial o_{11}} * \frac{\partial o_{11}}{\partial w_{11}}$$

$$o_{31} = \sigma \left( \underbrace{w_3 * o_{21}}_z + b_3 \right) \quad z = w_3 * o_{21} + b_3$$

$$o_{31} = \sigma(z)$$

$$\frac{\partial o_{31}}{\partial o_{21}} = \frac{\partial (\sigma(z))}{\partial (z)} * \frac{\partial z}{\partial o_{21}} \quad \left\{ \text{Chain Rule} \right\}$$

$$\boxed{0 \leq \sigma(z) \leq 0.25} \quad * \quad \frac{\partial((w_3 + b_3) + b_3)}{\partial(w_{21})}$$

↓

Derivative of  
Sigmoid

$$\boxed{\frac{\partial o_{31}}{\partial o_{21}} = 0 \leq \sigma(z) \leq 0.25 * w_{3 \text{ old}}} \quad }$$

This causes  
loss of Gradient  
which means, we can't  
have updated values

- ① To fix this problem Researchers started exploring other  
Activation function  
 In this case, we will use Tanh instead of Sigmoid function.
- ④ Tanh ④ ReLU ④ PReLU ④ Swish

### Conclusion;

- Perceptron → No chain Rule
- multi Layered Neural Network ; we do use chain Rule of derivatives to update all the weights (composite fns).

### Steps ;

1. Inputs
  2. Weights
  3.  $\sum w_i x_i + \text{Bias}$
  4. Hidden Layers
  5. Activation functions
  6. Loss functions ( $y - \hat{y}$ )
  7. calculate derivatives for  $w/B$
  8. update  $w_{\text{new}} = w - \eta \frac{\partial h}{\partial w_{\text{old}}}$
- $B_{\text{new}} = B - \eta \frac{\partial h}{\partial B_{\text{old}}}$

Derivative		
Activation	Derivative Range	Max Derivative
ReLU	0 or 1	✓ 1
Leaky ReLU	Small positive to 1	✓ 1
Sigmoid	0 to 0.25	✓ 0.25
Tanh	0 to 1	✓ 1
Softmax	0 to 0.25 (per class)	✓ 0.25

Imp: Sigmoid value derivative  $\leq 0.25$  always

Q11 Difference b/w Sigmoid and Tanh

Tanh = 0 centered, -1 to 1, hyperbolic

Sigmoid = Not 0 centered, 0 to 1, sigmoid

V.V. Imp Derivative value for different Gradients.

Activation	Derivative Range	Max Derivative
ReLU	0 or 1	✓ 1
Leaky ReLU	Small positive to 1	✓ 1
Sigmoid	0 to 0.25	✓ 0.25
Tanh	0 to 1	✓ 1
Softmax	0 to 0.25 (per class)	✓ 0.25

Gradient Vanishing problem since low numbers.

Remember → Gradient vanishing can still occur in Tanh

### Evolution of ACTIVATION FUNCTIONS

1. Sigmoid / softmax
2. Tanh
- Best 3. ReLU → one problem  
↳ Dead Neuron when Negative
4. Leaky ReLU → solves problem of Dead Neurons, by leaking Negative values.