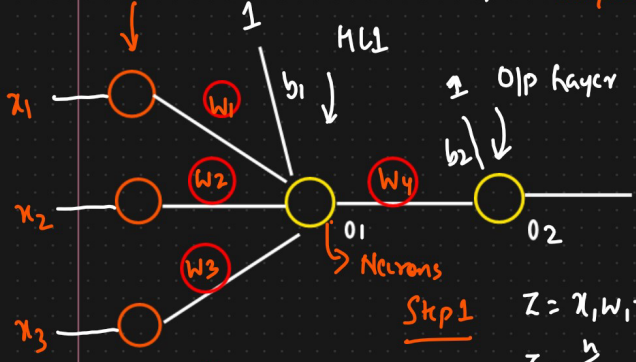
5 Multi layered Perceptron Model [Artificial Neural N/w]

① Forward Propogation
② Backward Propogation → Geoffrey Hinton →

③ Loss function
④ Optimizers ✓ Gradient Descent.
⑤ Activation function.

Multi layered NN

I/P layer — Forward Propogation → 2 layered Neural N/w



Backward Propogation.

Loss function = (1 − 0.51129) ≈ 0.49

x_1	x_2	x_3	Pass/Fail
IQ	Study hours	Play hours	O/p
95	4	4	1
100	5	2	1 ←
95	2	7	0

Error

Step 1 $z = x_1w_1 + x_2w_2 + x_3w + b_1$

$z = \sum_{i=1}^{n} w_i^T x_i + b$

① Forward Propogation 2: Activation(z)

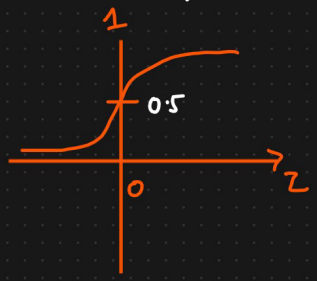
W_1 W_2 W_3 [0.01, 0.02, 0.03] [0.001]

Hidden Layer 1

Step 1: $z = 95 \times 0.01 + 4 \times 0.02 + 4 \times 0.03 + 1 \times 0.01$

$= 1.151$

Sigmoid. $= \frac{1}{1+e^{-z}}$



Step 2: Activation (z)

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

$O_1 = 0.759$

Hidden Layer 2

$W_4 = 0.02$

$b_2 = 0.03$

Step 1: $z = O_1 * W_4 + b_2$

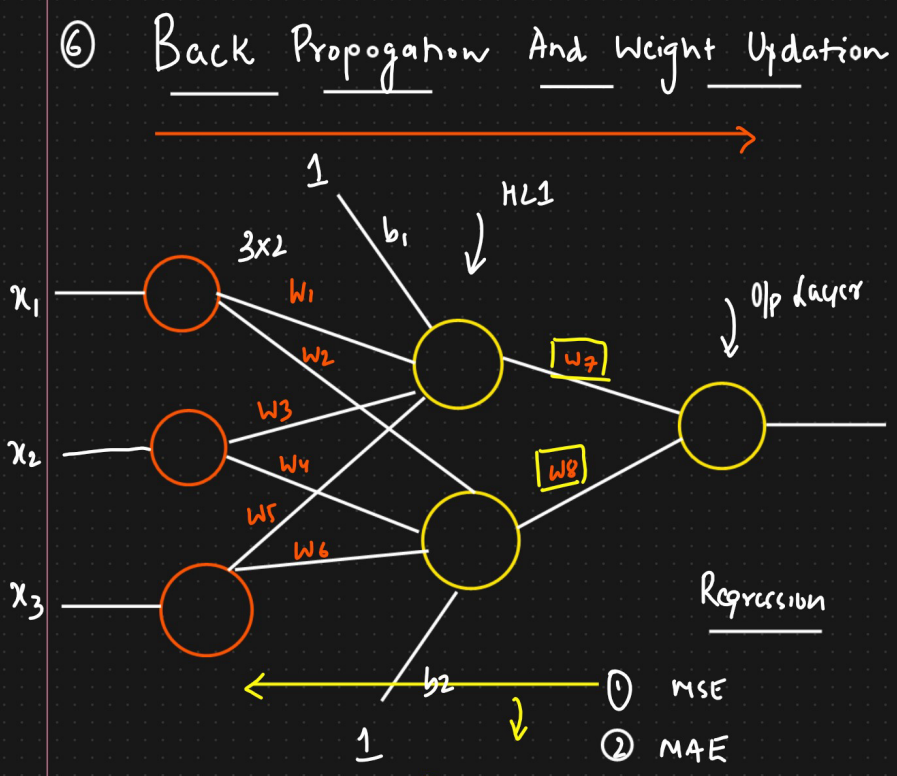
$= 0.759 * 0.02 + 0.03$

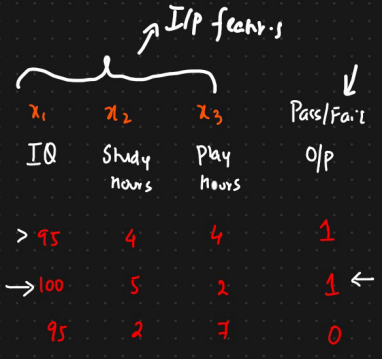
$= 0.045180$

Step 2 : Activation(z) $\frac{1}{1+e^{-(0.04518)}} = 0.511129$

$O_2 = 0.511129 \Rightarrow \hat{y}$

⑥ Back Propogation And Weight Updation Formula





Loss function $(y-\hat{y})^2$

Regression
① MSE
② MAE
③ Huber Loss

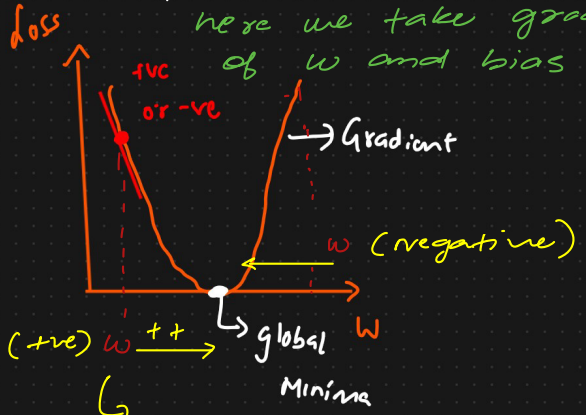
Classification
① Binary Cross Entropy
② Categorical CROSS Entropy

Weight updation Formula

similar to ML but here we take gradient of w and bias also

$$W_{7_{new}} = W_{7_{old}} - \eta \left[\frac{\partial L}{\partial W_{7old}} \right]$$ ← slope

$$W_{8new} = W_{8old} - \eta \left[\frac{\partial L}{\partial W_{8old}} \right]$$ ← Learning Rate



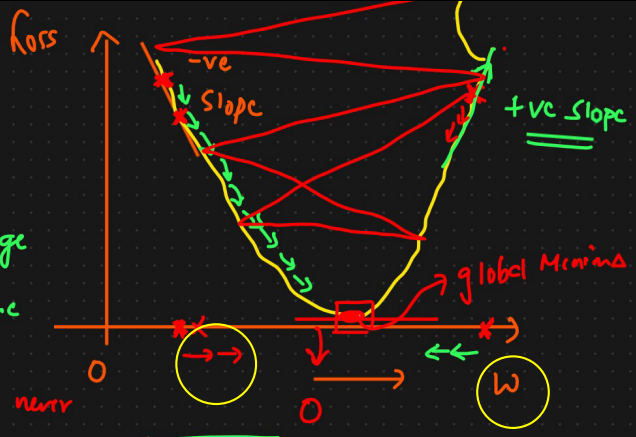
My aim is to increase weight to reach Global minima.

v.v.imp

$$W_{new} = W_{old} - \eta \frac{\partial L}{\partial W_{old}}$$ ⇒ Weight Updation Formula.

Gradient Descent ↗ Optimizers

Optimizers : To reduce the loss value



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

$= W_{old} + \eta\,(+ve)$

$W_{new} >> W_{old}$

η = large value ⇓ It may never converge

↳ learning Rate

0.001 → Small value.

↳ learning Rate

0.001 → standard.

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

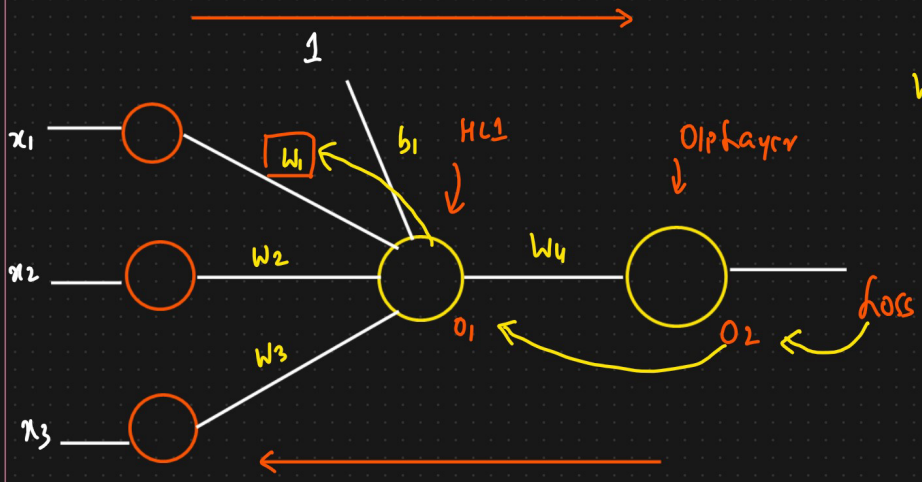
$= W_{old} - \eta\,(+ve)$

$W_{new} << W_{old}$

When W reaches global Minima

$W_{new} = W_{old}$

① Chain Rule of Derivative → used to find derivatives of composite functions (function inside a function).



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

$W_{4_{new}} = W_{4_{old}} - \eta \frac{\partial h}{\partial W_{4_{old}}}$

$$\frac{\partial h}{\partial w_{4old}} = \frac{\partial h}{\partial o_2} * \frac{\partial o_2}{\partial w_{4old}} \Rightarrow \text{Chain Rule of Derivation}$$

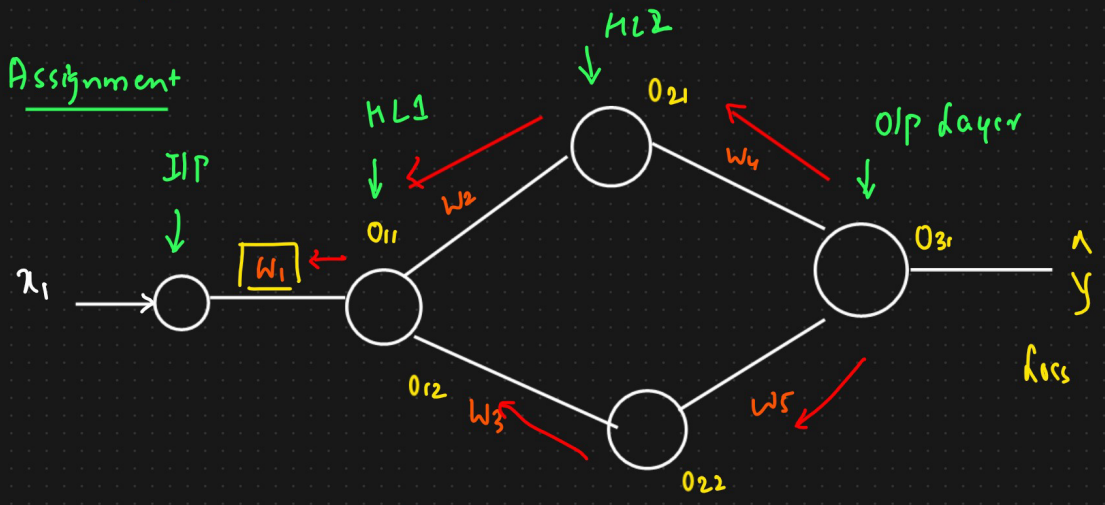
$$w_{1new} = w_{1old} - \eta \boxed{\frac{\partial h}{\partial w_{1old}}}$$

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

w_{2new}

w_{3new}.

Assignment



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

$$\Downarrow$$

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

$$+$$

$$\left[\frac{\partial h}{\partial o_{31}} * \frac{\partial o_{31}}{\partial o_{22}} * \frac{\partial o_{22}}{\partial o_{12}} * \frac{\partial o_{12}}{\partial w_{1old}}\right]$$

$\partial h \rightarrow \partial w_{old}$ (Chain Rule)

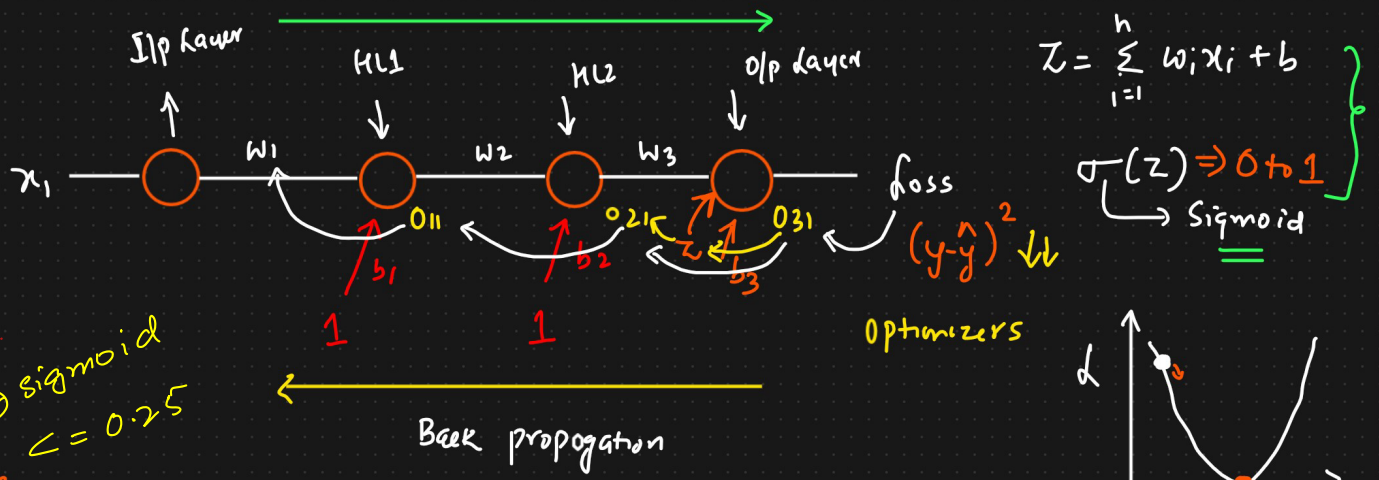
vvimp

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 $\phi(z)$



$$z = \sum_{i=1}^{n} w_i x_i + b$$

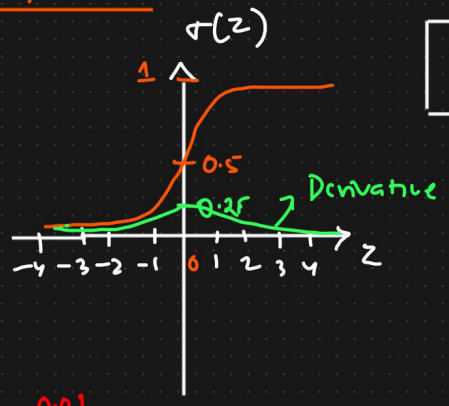
$\sigma(z) \Rightarrow 0$ to 1 → Sigmoid

Imp.
∂ sigmoid $< = 0.25$

Vanishing Gradient Problem.

⑨ Sigmoid Activation function

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



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

Derivative $(\sigma(z))$

$$0 \le \sigma(z) \le 0.25$$

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

$\eta = 0.01$

$\left[\frac{\partial L}{\partial W_{1\,old}}\right]$ ⇒ Small value

$\Rightarrow W_{1\,new} \approx W_{1\,old}$ (Equal)

$\left[\frac{\partial L}{\partial W_{1\,old}}\right]$ ⇒ Small Value

$\left[\frac{\partial L}{\partial W_{1\,old}}\right] = [0-0.25] * [0-0.25] * [0-0.25]$ ⇒ Smaller value (0.000001)

$$\frac{\partial L}{\partial W_{1\,old}} = \frac{\partial L}{\partial O_{31}} * \frac{\partial O_{31}}{\partial O_{21}} * \frac{\partial O_{21}}{\partial O_{11}} * \frac{\partial O_{11}}{\partial w_{11}}$$

$$O_{31} = \sigma(\underbrace{w_3 * O_{21} + b_3}_{z}) \qquad z = w_3 * O_{21} + b_3$$

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

$$\frac{\partial O_{31}}{\partial O_{21}} = \frac{\partial(\sigma(z))}{\partial(z)} * \frac{\partial z}{\partial O_{21}}$$ (Sigmoid) {Chain Rule}.

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

⇓

Derivative of Sigmoid

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

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

* To fix this problem Researchers started exploring other Activation function

In this case, we will use tanh instead of Sigmoid function.

① Tanh ② Relu ③ Prelu ④ Swiss

Conclusion;

- Perceptron → No chain Rule (single layer) No hidden layer
- MLP Multi Layered Neural Network; we do use chain Rule of derivatives to update all the weights (composite fxs).

Steps;

1. Inputs
2. weights
3. $\sum w_i \times x_i + Bias$
4. Hidden Layers → Act fn → output
5. Activation functions
6. Loss functions $(y - \hat{y})$
7. Calculate derivatives for w/B
8. Update $w_{new} = w - \eta \, \partial h / \partial w_{old}$

   $B_{new} = B - \eta \, \partial h / \partial B_{old}$

} Chain Rule of Derivatives

Imp. Sigmoid value derivative ≤ 0.25 always

Derivative

ELU → more time / complex

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

Q. 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 → 0.25
2. Tanh
3. ReLU → One problem
   ↳ Dead Neuron when Negative
4. Leaky Relu → Solves problem of Dead Neurons, by leaking Negative values.

Best