Deep Learning

→ Scientists Wants to invent a
→ A machine Which learns and do things similar to human.

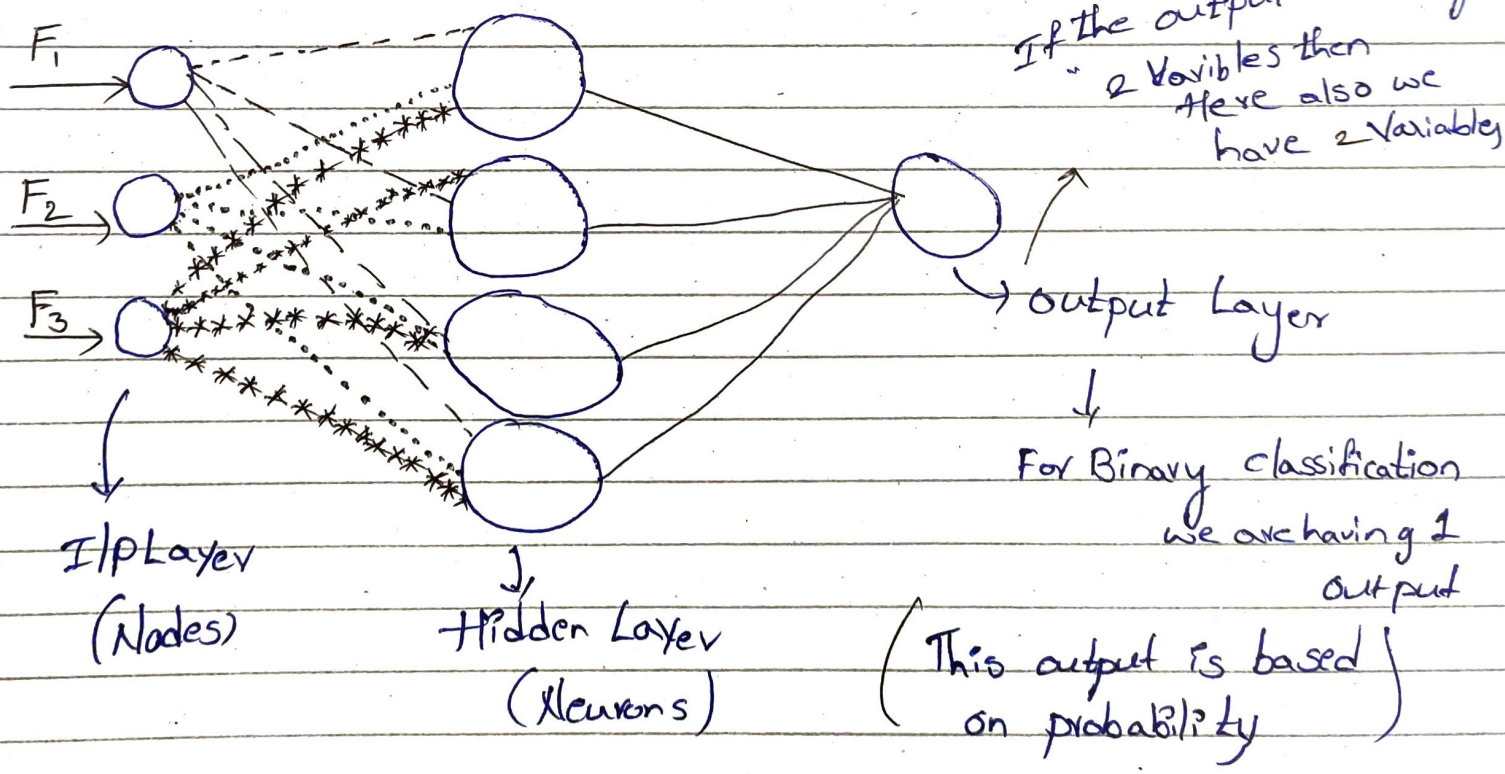
↓

For this Scientists created Neural Networks

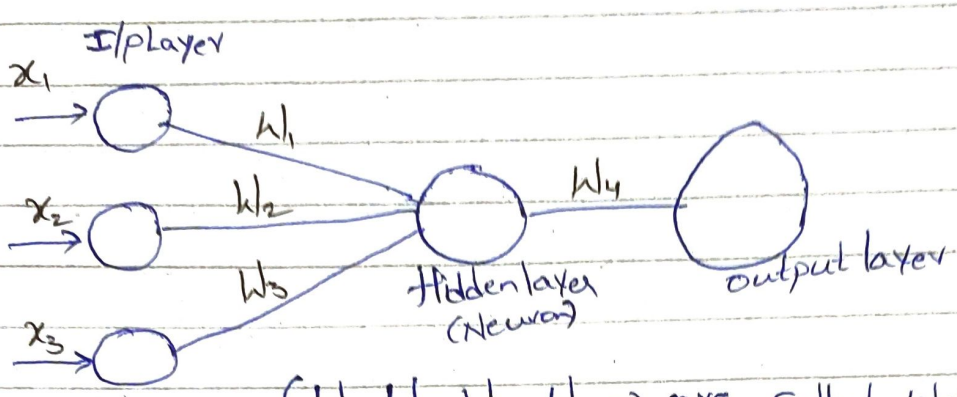
1st Simple type of Neural Network is called perceptron

→ This perceptron is having some issues then Jeffry Hinton discovered / created Back propagation.

Simple Neural Network Architecture! →



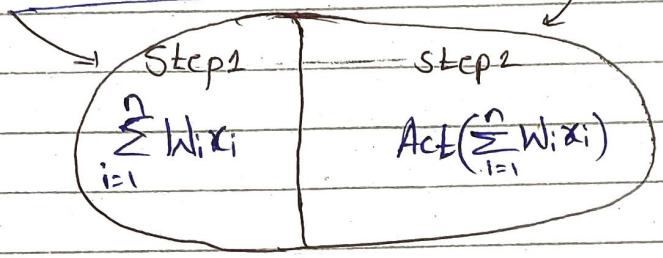
How the Neural Network Works.



(w_1, w_2, w_3, w_4 → are called weights)

In the hidden layer (Neuron)

↳ there are 2 parts



Ex:-

→ Suppose if i keep normal ~~brand~~ water on my hand then it won't affect.

→ Suppose if i keep very hot water on hand then it will immediately neurons sends information.

→ Here in the both examples neurons are transmitting information but this weights are making impact.

→ When i kept hot water then the weights might be high & immediately sends information.

In the Neuron

Step-1:

→ In the Neuron 1st step

$$Y_1 = W_1X_1 + W_2X_2 + W_3X_3$$

adding bias

$$Y = W_1X_1 + W_2X_2 + W_3X_3 + \text{bias}$$

{ Each feature is multiplied by weights

$$Y = \sum_{i=1}^{n} W_iX_i + \text{bias}$$

Step-2:

apply activation Function.

$$Z = Act(Y)$$

At the output:-(z)

$$z = Z \times W_4$$

$$= Act(Z)$$ → So here again we applied Activation Function.

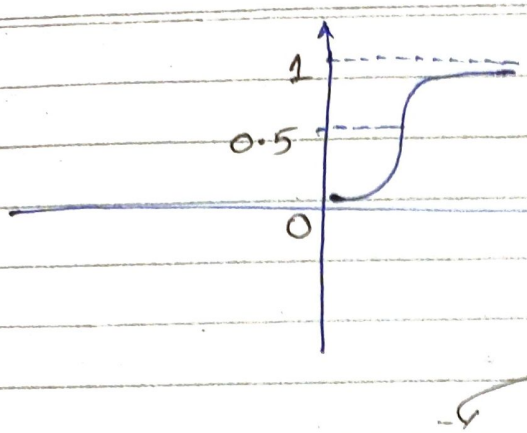
Activation Functions

1. Sigmoid Function:- $\frac{1}{1+e^{-y}}$ → Transform Values -1 to 1

In simple terms, like when you kept normal water on hand then the neuron won't sends information to brain.

But when you kept hot water neuron will be activated and passes information to brainquickly.

In this case neuron will be deactivated.



$y = \sum_{i=1}^{n} w_i x_i + bias$

$\rightarrow f(y) = \frac{1}{1+e^{-y}}$

If the Value $< 0.5 \longrightarrow 0 \rightarrow$ neuron not activated

If the Value $> 0.5 \longrightarrow 1 \rightarrow$ neuron activated.

{We will Consider only when > 0.5
$0.5 \rightarrow$ is the threshold}

2. Relu Activation:-

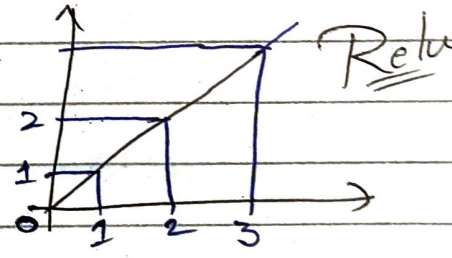
$Max(y, 0)$

$y = w_1x_1 + w_2x_2 + w_3x_3 + bias = \sum_{i=1}^{n} w_i x_i + bias.$

→ If $y = +ve$ then $Max(+ve, 0) = +ve \Rightarrow y$

→ If $y = -ve$ then $Max(-ve, 0) = 0 =$

→ This will always gives me maximum Value.



Suppose you are solving classification problem then in the Hidden layer we will use Relu and in the output layer we use Sigmoid Function (Because we applied Activation Functions in the Hidden & output

Basically why we prefer Sigmoid at the output layer is because Sigmoid will classify the output nicely because it has values 0 to 1. (According to the probability we can choose).

(and you know Sigmoid will treat the outliers problem also)
(Converted to same range 0 to 1

Neural Network Training

→ I am having dataset which contains 3 columns & 1 output

Play x_1	study x_2	sleep x_3	output y
2h	4h	8h	1
⋮	⋮	⋮	⋮

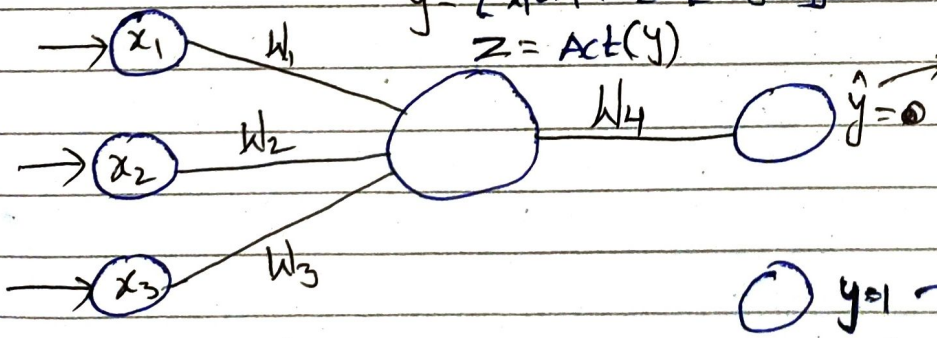
output → 1 (or) 0
↓ pass ↓ Fail

Input → play, study, sleep
No. of hours he is playing (or) studying (or) sleeping.

→ So My neural Network has 3 nodes(input) & 1 output layer.

$y = [x_1w_1 + x_2w_2 + x_3w_3] + bias$

$z = Act(y)$



○ $y_{=1}$ → Actual output

So, in this example, i have predicted my output $\hat{y}=0$ but my output is $y=0$

↳ I need to Compare my predicted output with the Actual output

↳ For that i need Loss Function

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

Q) Why Loss Function is having Square

A) suppose if the Loss Function = $y-\hat{y}$

Case(i) $y=0$, $\hat{y}=1$

$\Rightarrow$ Loss $= (0-1)$

$= -1$

Case(ii) $y=1$, $\hat{y}=0$

$\Rightarrow$ Loss $= 1-0$

$= 1$

→ Here we are getting Negative, thats why are eliminating Negative by using Square

Q) What if you have multiple outputs

Ans: Then i will Sum it up

$$\Rightarrow \sum_{i=1}^{n} (y-\hat{y})^2 \Rightarrow \text{Cost Function}$$

It won't be called as Loss Function

→ Called as Cost Function.

→ So, we always want to have Minimize Loss Function

→ Because, when my outputs are Correct then my Loss Function is zero

→ If my outputs are not Correct then my Loss Function is 1

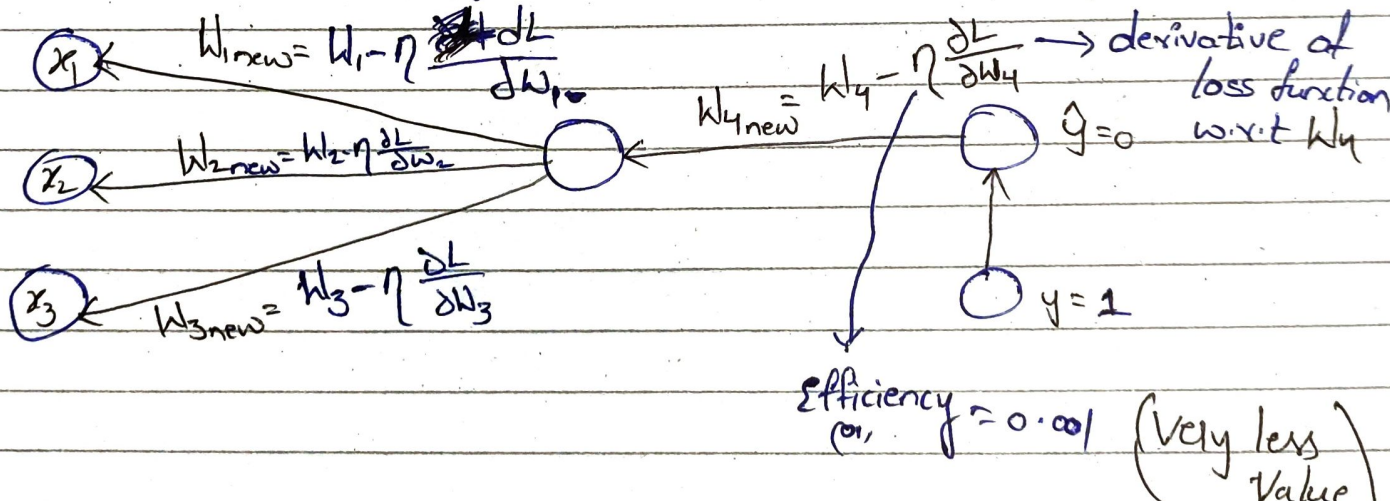
$\Rightarrow (y-\hat{y})^2 = (1-0)^2 = \underline{1}$

→ Thats why we always wants to minimize loss function

→ Such that actual output and predicted o/p should be Same.

↳ In order to reduce loss function we need optimizer.

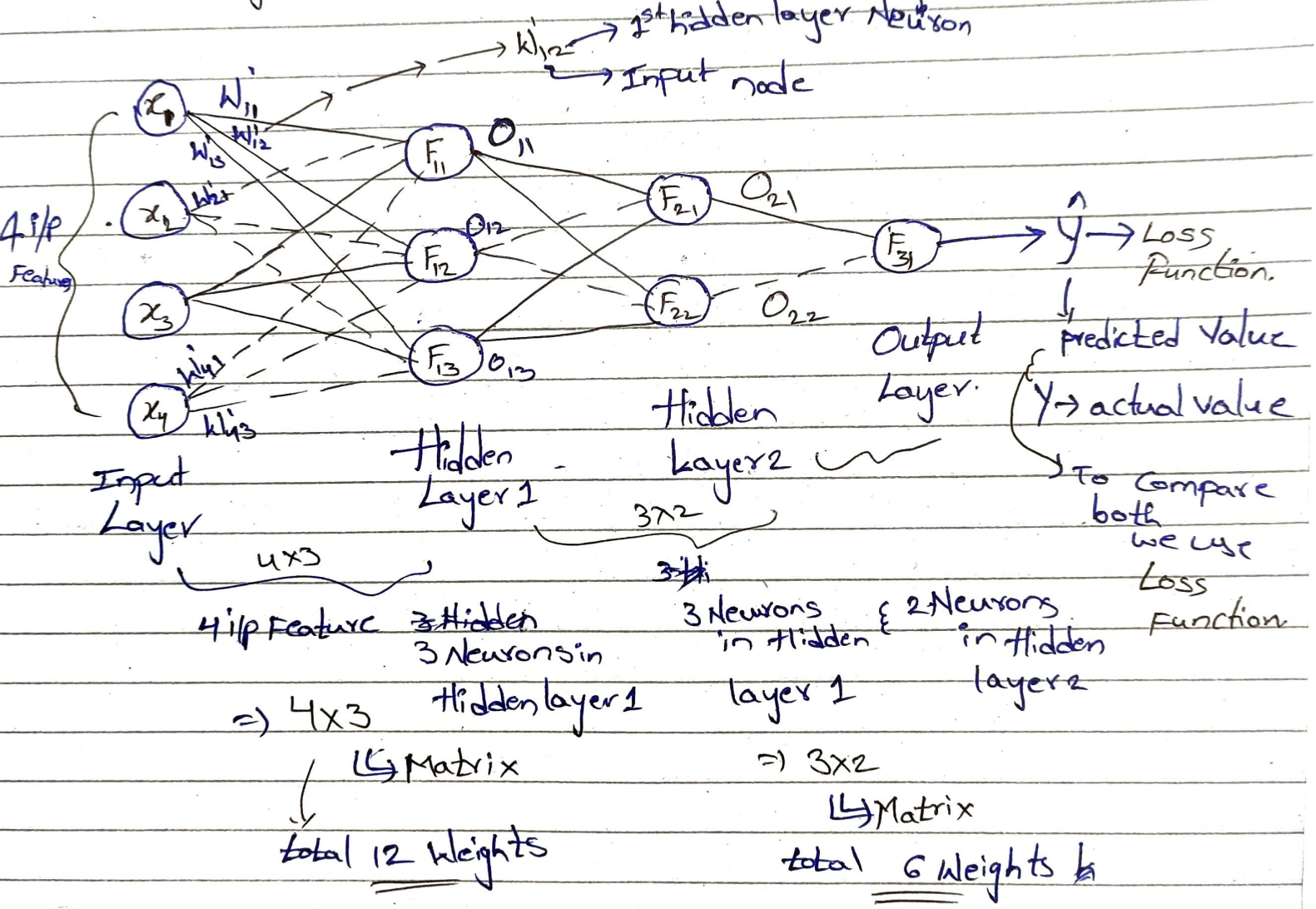
↳ This optimizer will helps us in adjusting weights in in Back propagation.



(Here w_1, w_2, w_3, w_4 → old
↳ using optimizer we are updating $w_1, w_2, \ldots w_4$ to new weights.

↳ After the weights are adjusted we need to do again Forward propagation. This process will go until some number of Epochs or the loss value completely reduced.

Multi layer Neural Network (MNN)



Note: ① $O_{11}, O_{12}, O_{13}, O_{21}, \ldots, O_{22}$ → all these are outputs.

② In the First hidden layer we wrote F_{11}, F_{12}, F_{13} → all these will apply activation Functions. (Lets say $w_{11}x_1 + \ldots + w_{43}x_4 = y$ on this we will apply activation Function (Y_1) we get O_{11}.

Loss Function:→

↳ $(y-\hat{y})^2$

→ We want to reduce the loss function.

↓

In order to reduce we use optimizer to update the weights.

↳ (Back propagation)

Gradient Descent

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

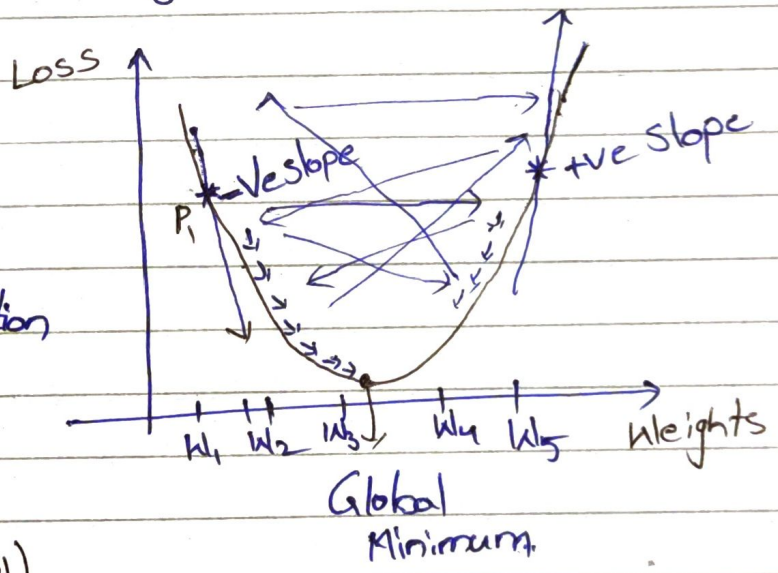
$\frac{\partial L}{\partial W_{old}}$ → loss Function

η → Learning Rate

derivative of Loss Function w.r.t to W_{old}

→ Derivative is nothing but Slope.

→ We want to reach Global minimum then only our Loss Function will be less.



η → very less (0.001)

⇒ Suppose My loss value is at P_1 then i calculated slope using derivative & it is −ve (pointing down)

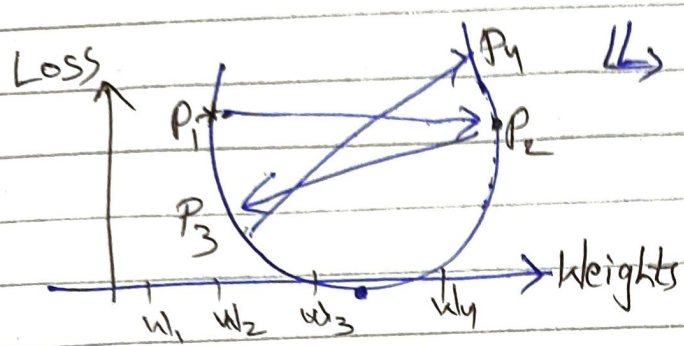
⇒ My equation becomes $W_{new} = W_{old} - \eta(-ve) = W_{old} + \eta(slope)$ (some −ve value)

⇒ So, My weight is increasing

→ So, here my weight is increasing at a speed of η

η ⇒ Very high.

⇒ If i keep $\eta = 10000$ ⇒ then my weight increases rapidly



↳ $w_{new} = w_{old} - (1000)(-\text{slope})$

(Considered P_1 ↓ -ve slope)

⇒ $w_{new} = w_{old} + 1000(\text{slope})$ ↓

In this case we never reach global minimum

↳ Its like weights will be updated from

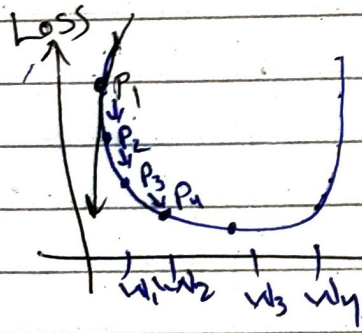
$P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_4$.

↳ So it will never Converge.

→ η = Very low.

↳ then my weight moves very slowly

$\eta = 0.001$ ↳ $w_{new} = w_{old} - (0.001)(-\text{slope})$



⇒ $w_{new} = w_{old} + 0.001\text{slope}$

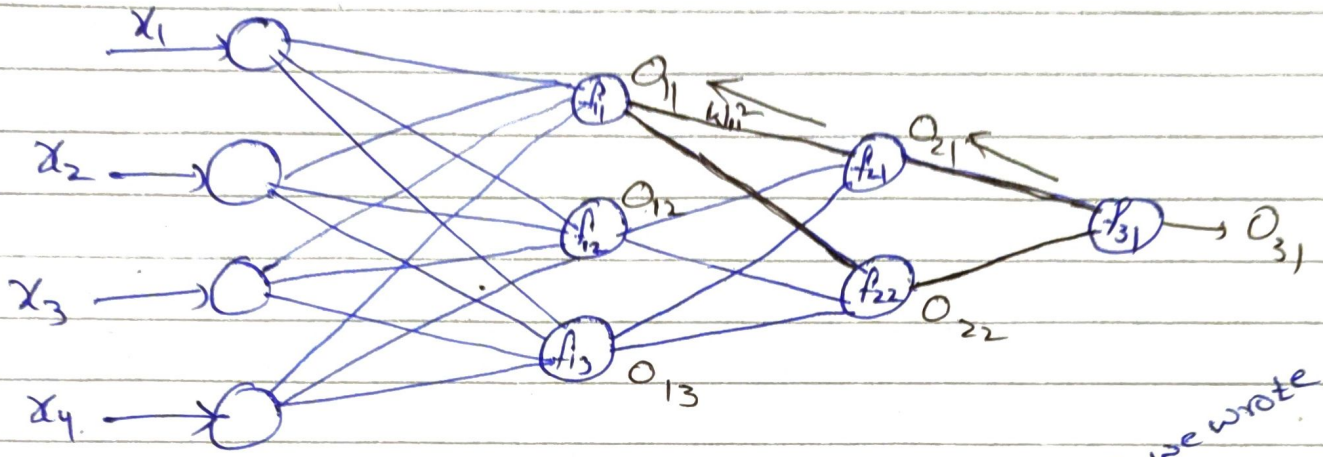
⇒ w_{new} increased at very low pace/speed.

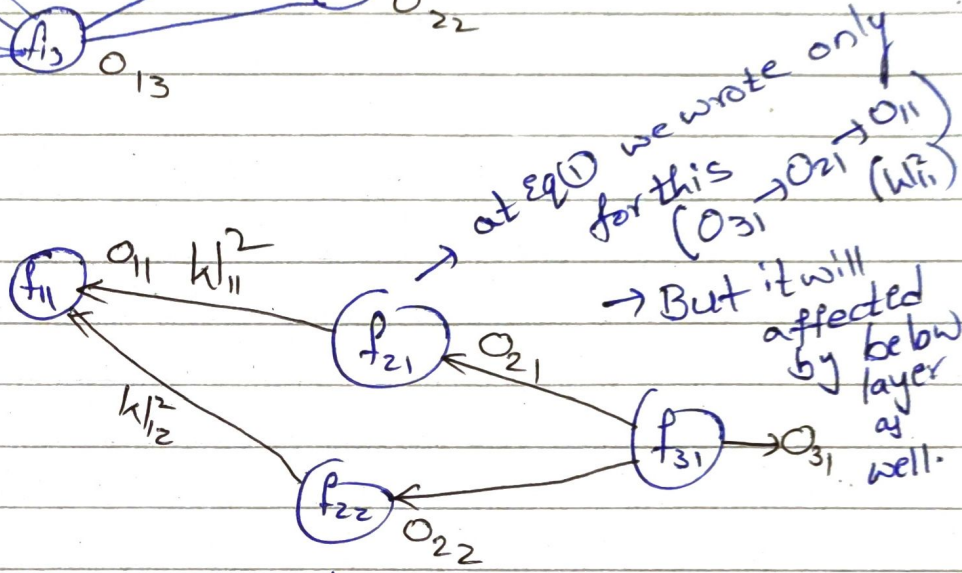
→ So here weights will be updated slowly

~~not~~ like $(P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_4)$

⇒ at it will reach Global Minimum (Converges)

at Eq① (Hidden 2 to Hidden 1)





$$\frac{\partial L}{\partial W_{11}^2} = \frac{\partial L}{\partial W_{11}^2}$$

Here in both ways O_{11} is affecting. So, we need to add both

$$\frac{\partial L}{\partial W_{11}^2} = \left[\frac{\partial L}{\partial O_{31}} \times \frac{\partial O_{31}}{\partial O_{21}} \times \frac{\partial O_{21}}{\partial W_{11}^2}\right] + \left[\frac{\partial L}{\partial O_{31}} \times \frac{\partial O_{31}}{\partial O_{22}} \times \frac{\partial O_{22}}{\partial W_{12}^2}\right]$$

upper — lower.

→ until we reach global minimum, this weights will be updated in Back propagation (or) it will do for some epochs.

→ Epochs! → 1 Forward & 1 Back propagation ⇒ i/p ⟶ ⟵ o/p (1 Epoch)
↳ is called 1 epoch.