__/__/____

Q) why Deep learning is too much famous?

→ 1) Applicable in each and every domain
2) able to solve vast amount of problem.
3) DL model can automatically learn relevant feature from the data, reducing the need for manual feature engineering.
- DL models can scale effectively with large amounts of data and computational resources.

* feature extraction in deep learning
- feature extraction refers to the process of automatically learning and extracting relevant features or pattern from raw data
- Unlike traditional machine learning approaches that often relay on manual feature engineering, deep learning model can automatically learn intricate representations directly from the input data.
- feature Extraction is critical step in deep learning that allows the model to understand and interpret Complex patterns within the data.

* deep learning Model →
1) Image classification :→ - Rest Net
- google Net

2) object detection :→ - YoLo
- SSD
- fast and faster RCNN

3) object segmentation → 1) Mask RCNN
- U-Net
- V-Net

4) Image translation → - Pix to Pix

5) Speech generation → - Webnet

6) Text classification → - BERT

* Transfer learning :-

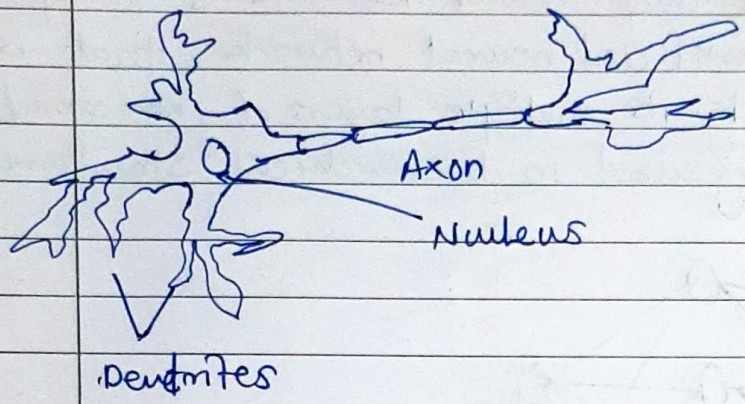
- In transfer learning a pre-trained model, often trained on a large dataset for a related task, serves as a starting point for training new model.
- Instead of initializing the new model from scratch, the weight and parameters of the pre-trained model are utilized as foundation.

• ways to apply transfer learning in deep learning:

① feature extraction :
- This envolve using pre-trained model as a fixed feature extractor. the earlier layers of the pre-trained model are used to extract relevant features from the input data, and this features are then fed into a new, custom classifier or set of layers tailored to specific task

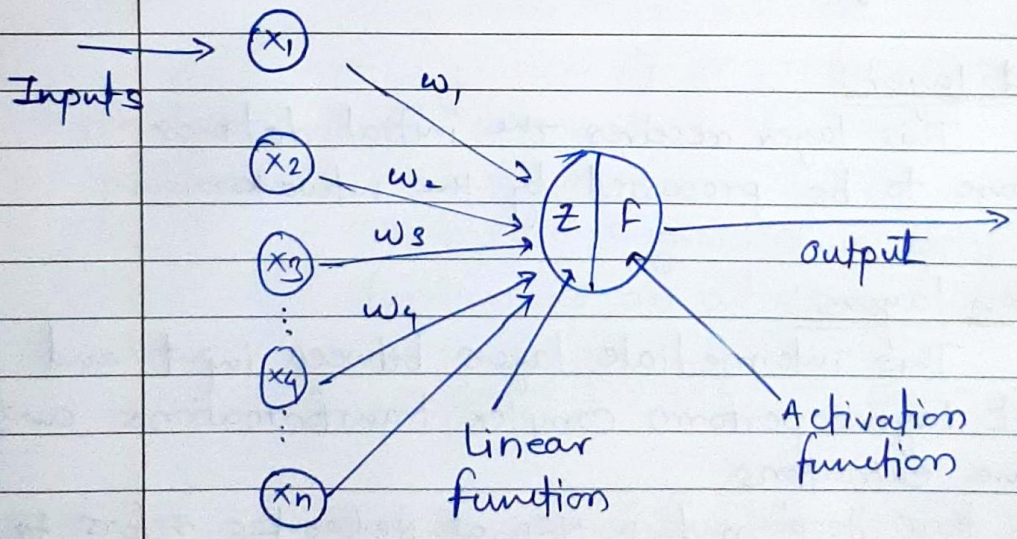
② fine-tuning :- The pre-trained model's are further adjusted or fine-tuned during the training process on the new dataset. the goal is to allow to adapt its learned representations to the nuance of the new data while still retaining the knowledge gained from the original task.

* Artificial Neuron :- (Perceptron)



Human Neuron.

Artificial Neuron Inspired By Human neuron.

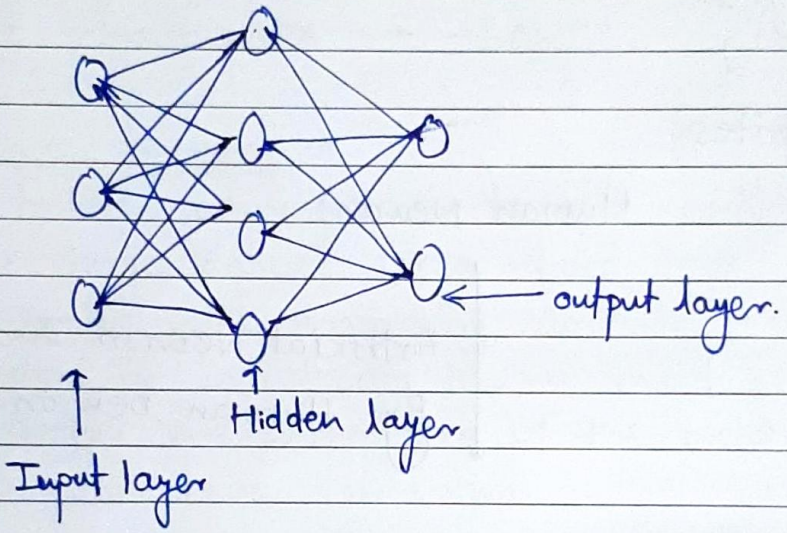


Artificial Neuron [Perceptron]

__/__/____

* DNN → Deep Neural Network

- Basic unit of deep Neural Network is perceptron.
- is type of artificial neural network, that is characterized by its multiple layers of interconnection of neuron organized in hierarchical structure.



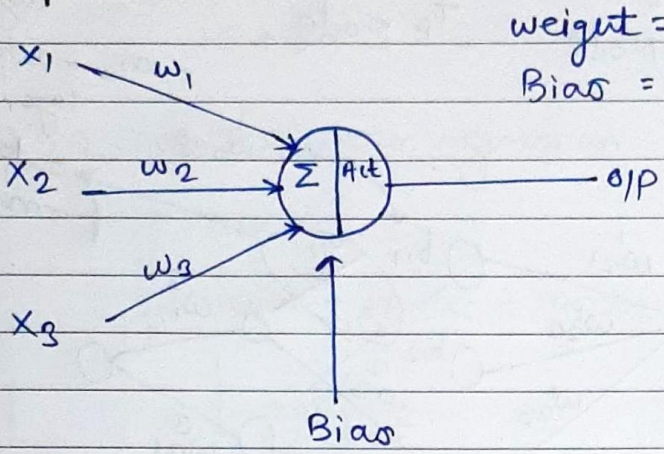
* Input layer :-
This layer receives the initial data or feature to be processed by the network.

* Hidden layers:
This intermediate layers between input and output layers perform complex transformations and feature extractions
- The term 'deep' in deep Neural Networks refers to the presence of multiple hidden layers.

* output layers :- produce final output of the network.

__/__/____

✱ perceptron :→



$$x_1w_1 + x_2w_2 + x_3w_3 + b \Rightarrow w^Tx + b$$

$$Act(w^Tx + b) \Rightarrow o/p$$

Similar to $y = mx + c$

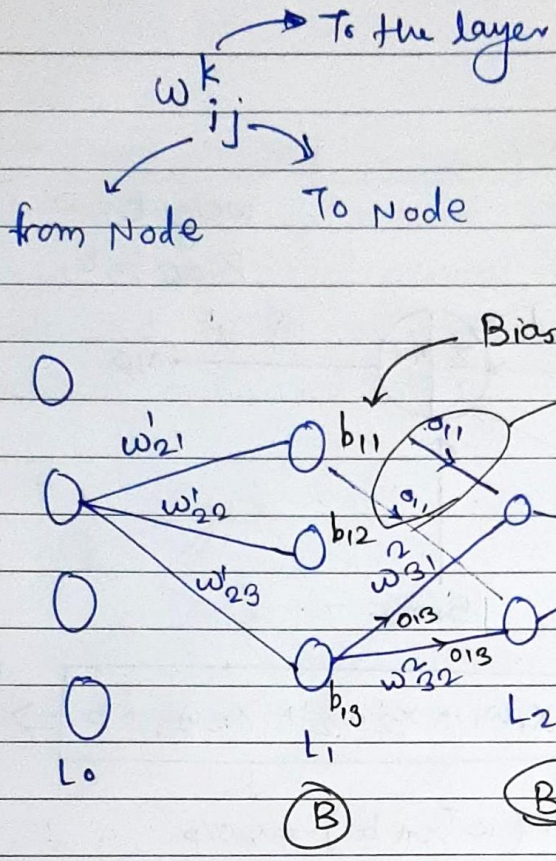
$$y = m_1x_1 + m_2x_2 + m_3x_3 + c$$

for classification problem ↓

In logistic Regression we apply sigmoid function as an activation function.

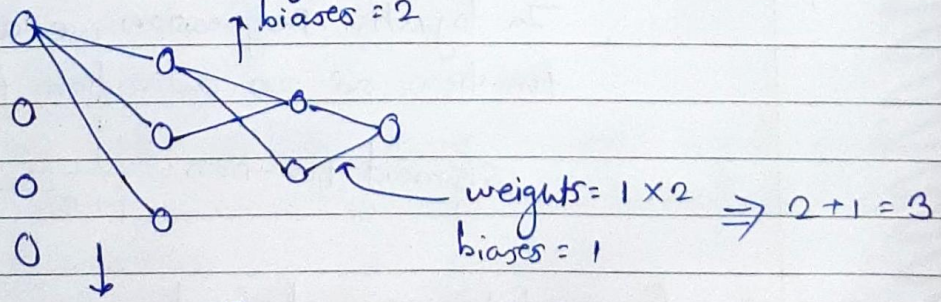
∴ sigmoid function = $\frac{1}{1+e^{-x}} = \frac{1}{1+e^{-(y)}}$

- if we take sigmoid function as our activation function then our perceptron above become similar to logistic regression.



- with respect to every perceptron we have bias, it have only to hidden layers and output layers
- Biases are Constant.

* Trinable parameters:



weights = ~~12~~ 4×3 = 12

biases = 3

4×3+3 = 15

∴ Total tinable parameter =

$15 + 8 + 3 = 26$

— Now, in case of linear regression.

$$y = mx + c$$

$$y : m_1x_1 + m_2x_2 + m_3x_3 + m_4x_4 + c$$

↓ optimizer.

The parameter going to change or update for greater accuracy will be (getting minimum loss.)

→ m_1, m_2, m_3, m_4, c

∴ these 5 parameter are tinable parameter.

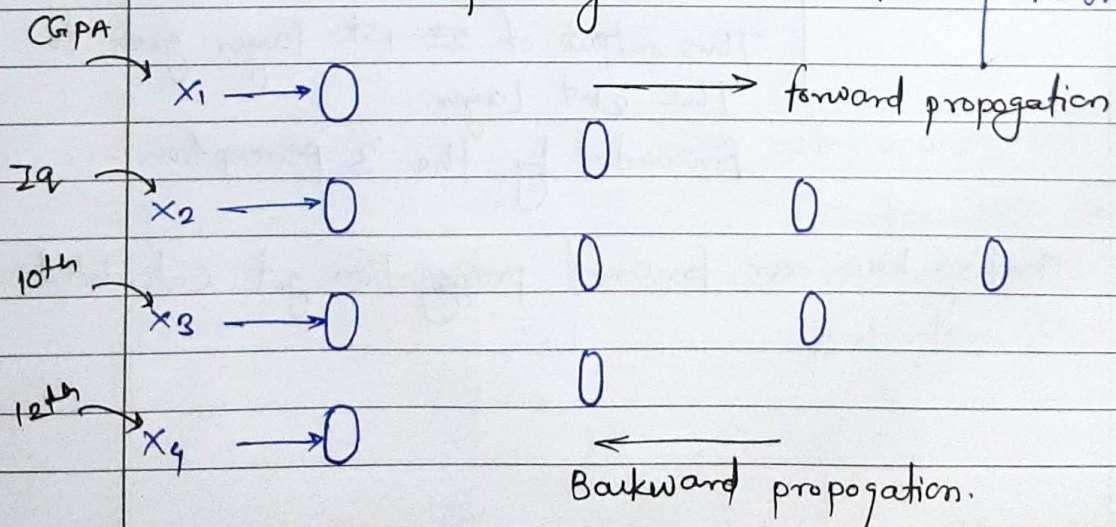
* Forward propogation :→

Data :

CGPA	iq	10th	12th	Placed
-	-	-	-	-

Hyperparameter
- Hidden layer.
- No. of Neuron.

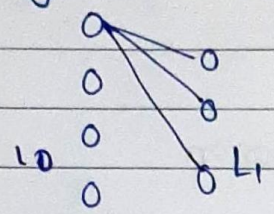
Input layer = No. of feature



__/__/____

* forward propgation calculation.

- for Layer 1 →



$$\begin{bmatrix} w'_{11} & w'_{12} & w'_{13} \\ w'_{21} & w'_{22} & w'_{23} \\ w'_{31} & w'_{32} & w'_{33} \\ w'_{41} & w'_{42} & w'_{43} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \end{bmatrix}$$

↓ Take the transpose

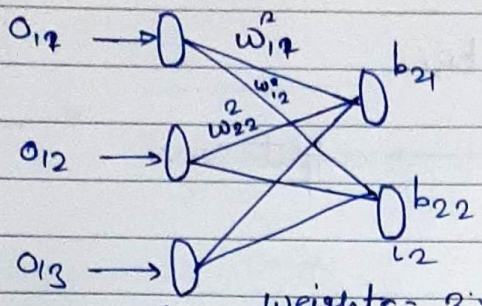
$$\begin{bmatrix} w'_{11} & w'_{21} & w'_{31} & w'_{41} \\ w'_{21} & w'_{22} & w'_{32} & w'_{42} \\ w'_{13} & w'_{23} & w'_{33} & w'_{43} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \end{bmatrix}$$

$$\begin{bmatrix} w'_{11}x_1 + w'_{21}x_2 + w'_{31}x_3 + w'_{41}x_4 \\ w'_{21}x_1 + w'_{22}x_2 + w'_{32}x_3 + w'_{42}x_4 \\ w'_{13}x_1 + w'_{23}x_2 + w'_{33}x_3 + w'_{43}x_4 \end{bmatrix} + \begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \end{bmatrix} = \begin{bmatrix} o_{11} \\ o_{12} \\ o_{13} \end{bmatrix}$$

↓ This output of ~~2nd~~ 1st layer goes to The 2nd Layer.
- forwarded by the 3 perceptron.

- That's how our forward propogation got calculated.

* for the 2nd layer



weights: $3 \times 2 = 6$

Biases: 2

$$\begin{bmatrix} w^2_{11} & w^2_{12} \\ w^2_{21} & w^2_{22} \\ w^2_{31} & w^2_{32} \end{bmatrix} \times \begin{bmatrix} o_{11} \\ o_{12} \\ o_{13} \end{bmatrix} + \begin{bmatrix} b_{21} \\ b_{22} \end{bmatrix} =$$

↓ Transpose

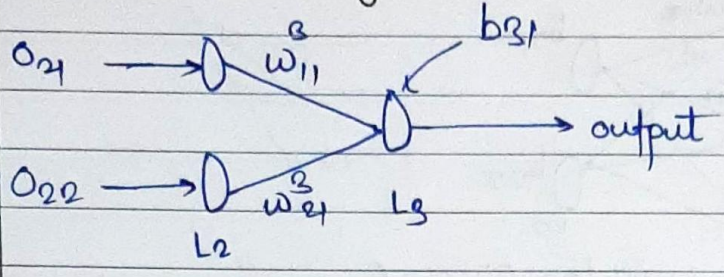
$$\begin{bmatrix} w^2_{11} & w^2_{21} & w^2_{31} \\ w^2_{12} & w^2_{22} & w^2_{32} \end{bmatrix} \times \begin{bmatrix} o_{11} \\ o_{12} \\ o_{13} \end{bmatrix} + \begin{bmatrix} b_{21} \\ b_{22} \end{bmatrix}$$

$$\left(\begin{bmatrix} w^2_{11} o_{11} + w^2_{21} o_{12} + w^2_{31} o_{13} + b_{21} \\ w^2_{12} o_{11} + w^2_{22} o_{12} + w^2_{32} o_{13} + b_{22} \end{bmatrix} = \begin{bmatrix} o_{21} \\ o_{22} \end{bmatrix} \right)$$

after applying activation ~~function~~ then you'll get the output for that particular layer.

__/__/____

* for the final layer



$$\left(\left[o_{21} \times w^3_{11} + o_{22} \times w^3_{21} \right] + b_{31} \right)$$

↓ activation function

final output → o_{31}

- final output may be prediction, classification can be anything w.r.t. the problem statment.

* loss function

- difference between loss and cost :-

Height	weight	BMI	BMI_Predicted
170	65	22	21
185	68	23	2.5
175	71	21	0.5

→ loss = 22 - 21 = 1

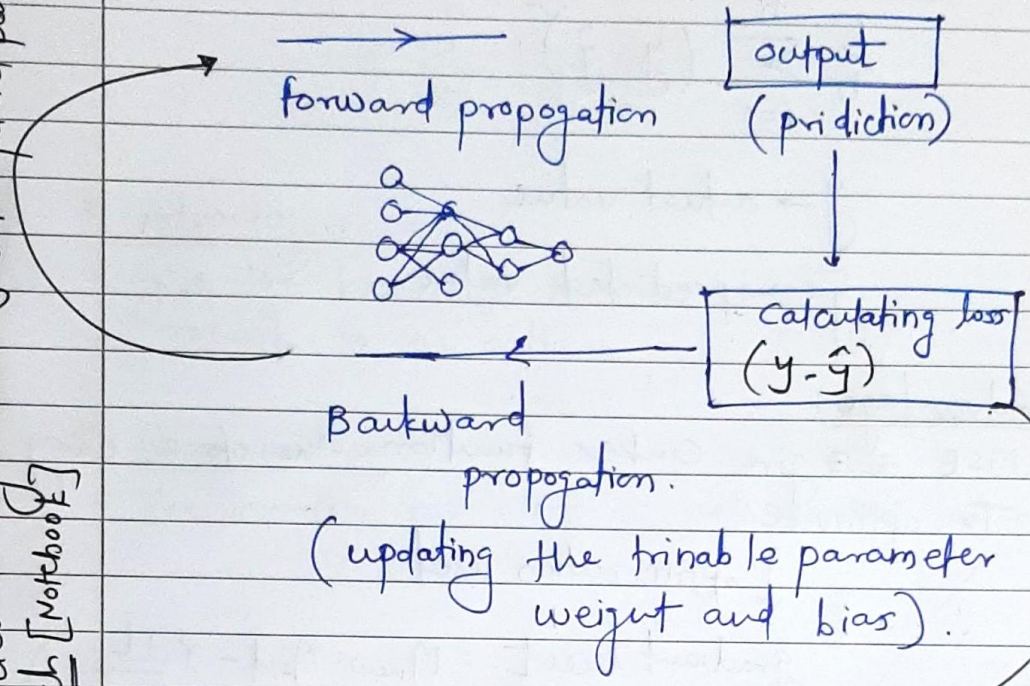
⇒ actual - predicted

- loss will take only one sample data in account at a time.

$$\text{Cost} = \frac{1}{n} \sum_{i=1}^{n} (\text{actual} - \text{Pred})$$

- cost will take whole data into consideration.

- **Loss function:**



The number of cycle decided by epoch [Notebook E]

again forward propogation with updated parameter.

∴ the loss is like a feedback

- if the loss is more then we have to update The trinable parameter to Reduce the loss.

- **Loss fuction**

Regression	classification
① MSE	① Binary Cross entropy or logloss.
② MAE	② Categorical Cross Entropy
	③ Sparce Categorical Cross entropy.

__/__/____

* MSE → Mean Square Error.

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

y → actual value

$\hat{y}$ → predicted value.

+ Advantage:

- MSE ~~will~~ give Convex function. therefore easy To optimize.

↓ optimization used

Gradiant decent : $m_{new} = m_{old} - \alpha \frac{\partial L}{\partial m}$

α → ~~loss funct~~ learning rate

L → loss function.

- easy to interpret

+ disadvantage:

- Not robust to the outlier
- for real value we have to ~~go~~ again take root of the Same.

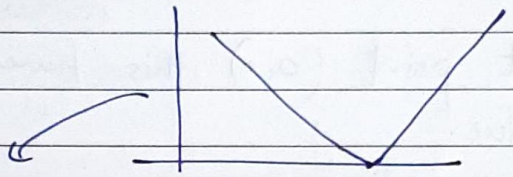
__/__/____

* **MAE :-** Mean Absolute Error.

$$\frac{1}{n}\sum_{j=1}^{n}|y-\hat{y}|$$

* **Advantage :-**
- give the loss in particular range.
- robust to the outliers.

* **Disadvantage :-**
- The function for MAE is not convex function



  - Non-differential
  - we can't use gradiant descent to optimize and find the global minima.

* **Backorder Propogation :-**
- The goal is to minimize the loss by adjusting the weights of neural network.
- Backorder propogation is a process of updating the weights by propogating the error backword through the network.

* Calculation of minima and maxima.

$z = (x^2 + y^2)$

Condition for Calculating Minima

$$\frac{\partial z}{\partial x} = 0 \quad \text{and} \quad \frac{\partial z}{\partial y} = 0$$

$$\frac{\partial (x^2+y^2)}{\partial x} = 0 \quad \text{and} \quad \frac{\partial (x^2+y^2)}{\partial y} = 0$$

$$2x = 0 \qquad 2y = 0$$

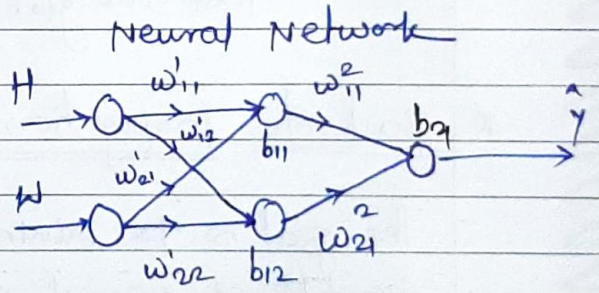
$$\boxed{x = 0} \quad \text{and} \quad \boxed{y = 0}$$

∴ at point (0,0) this function 'z' have minimum value.

* Backword propogation :->

understand the Backorder propogation and optimization of weight and Bias with example

Height	weight	BMI
160	60	21
170	65	22
180	70	20
190	71	18



Trinable parameter → 4+2+2+1 = 9

$\hat{y}$ depends upon this 9 parameter.

* **Step-1**
  - Initialize weight and bias
    Method → Random and define

* **Step-2**
  point (row)

* **Step-3** :-
  forward propogation
  [Matrix Multiplication, dot product]
  - output calculated $\hat{y}$.

* **Step-4**
  loss function
  suppose. $L = (y - \hat{y})^2$ → variable
  ↓ fix value

  let $\hat{y} = 24$

  $Loss = (21 - 24)^2$

  $\boxed{loss = 9}$

- our main aim to Reduce the loss

* **Step-5**
  gradiant decent [updating the weight and biases]

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

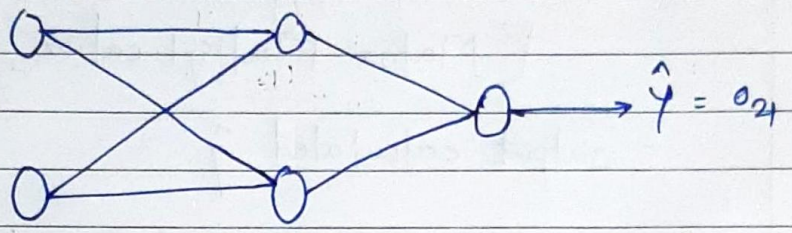
[Python → Impliment Backpropagation → code → from Scratch]

__/__/____

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

Loss function.

* Chain Rule of derivation :→



$$\hat{y} = o_{11} \times w_{11}^2 + o_{21} w_{21}^2 + b_{21}$$

$$Loss = y \cdot \hat{y}$$

for updating the w_{11}^2

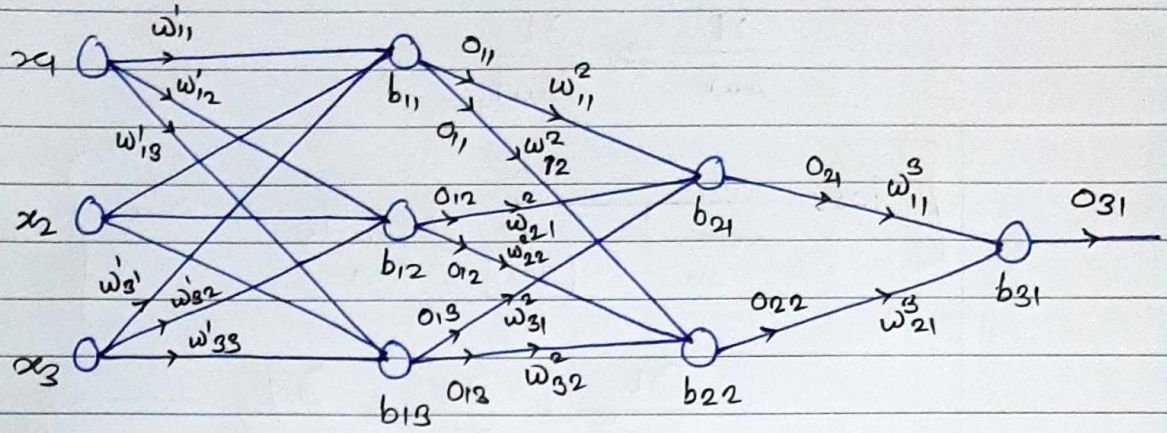
$$\frac{\partial L}{\partial w_{11}^2} = \frac{\partial L}{\partial \hat{y}}$$

$$= \frac{\partial L}{\partial o_{21}} \times \frac{\partial o_{21}}{\partial w_{11}^2}$$

$\hat{y}$ depends on w_{11}^2 depends on o_{21}

chain rule of derivation.

updating trivable parameter with backpropogation :→



Trinable parameter →

$3\times3+3=12$	$(3\times2)+2=8$	$(2\times1)+1=3$	Total 23

for updating :→ gradiant descent

Suppose we want to update 'w' [weight]

Then -

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

→ loss funtion

↓ learning Rate

Now,

will focus on only $\frac{\partial L}{\partial w}$

↓

and How chain Rule of derivation work.

take $w = w^3_{11}$

$$\frac{\partial L}{\partial w} \Rightarrow \frac{\partial L}{\partial w^3_{11}}$$

$$\left[\text{loss} \xrightarrow[\text{on}]{\text{depends}} \hat{y} \; [o_{31}] \xrightarrow[\text{on}]{\text{depends}} w^3_{11} \right]$$

$$\therefore \frac{\partial L}{\partial w^3_{11}} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial w_{11}{}^3} \quad \text{—— by chain Rule of derivation}$$

Now,

$$\left[w^3_{11} \xrightarrow[\text{on}]{\text{depends}} o_{21} \xrightarrow[\text{on}]{\text{depends}} w^2_{11} \right]$$

can include w^3_{11}

$$\left\{ \therefore \frac{\partial L}{\partial w^2_{11}} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial o_{21}} \times \frac{\partial o_{21}}{\partial w^2_{11}} \right.$$

Now,

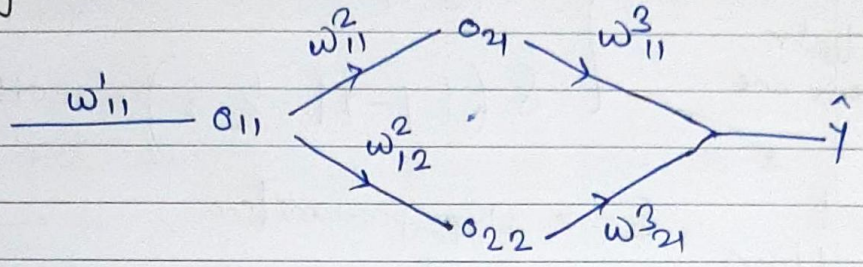
$$\left[\text{loss} \longrightarrow w^3_{11} \longrightarrow o_{21} \longrightarrow w^2_{11} \longrightarrow o_{11} \longrightarrow w'_{11} \right]$$

again can include weight which was Not included.

$$\frac{\partial L}{\partial w'_{11}} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial o_{21}} \times \frac{\partial o_{21}}{\partial w^2_{11}} \times \frac{\partial w^2_{11}}{\partial o_{11}} \times \frac{\partial o_{11}}{\partial w'_{11}}$$

* ~~The more, $\hat{y}$ we depend on weights~~

$\hat{y}$ → depend on two path.



mention the weights if you want to.

$\therefore \frac{\partial L}{\partial w^1_{11}}$ ⇒ will take both path into consideration

→ final output will be addition of two path.

$$\therefore \frac{\partial L}{\partial w^1_{11}} = \frac{\partial L}{\partial \hat{y}} \left[\frac{\partial \hat{y}}{\partial o_{21}} \times \frac{\partial o_{21}}{\partial o_{11}} \times \frac{\partial o_{11}}{\partial w^1_{11}} + \frac{\partial \hat{y}}{\partial o_{22}} \times \frac{\partial o_{22}}{\partial o_{11}} \times \frac{\partial o_{11}}{\partial w^1_{11}} \right]$$