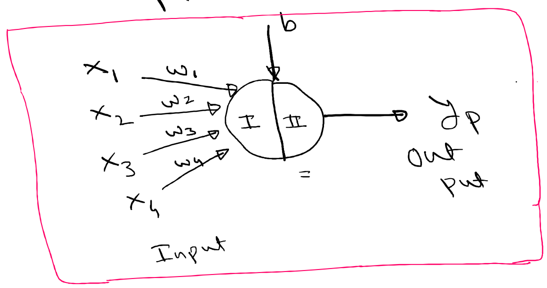
Deep learning Day-2

* Neuron → Perceptron → it is a two stage algorithm

weight, bias → External parameter



Stage I → Summation fun^n (z)

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

Stage II — Activation fun^n (Sigmoid)

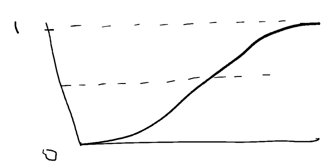
$$\boxed{\sigma(z) = \frac{1}{1+e^{-z}} = y_p}$$

This whole process called as perceptron and perceptron it self act as a Neuron in Neural Network.

* What is Sigmoid ?

→ it is used to add Nonlinearity.

$y = mx + c$ $\qquad$ $\sigma(y) = \frac{1}{1+e^{-y}}$



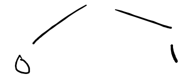
→ Sigmoid convert continous values into probability distribution.

→ Range → 0 to 1

→ it always give probabilistic value.

→ it always give probability value of any one class.

Class



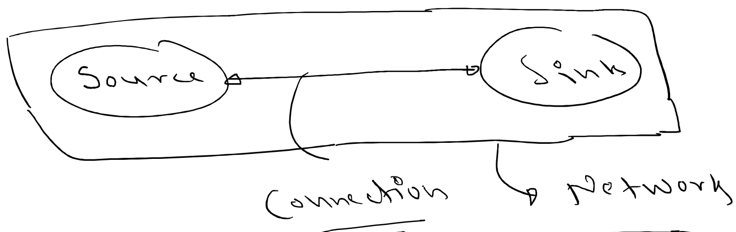
$$p(x \in y=1) = p(1) = 0.7$$

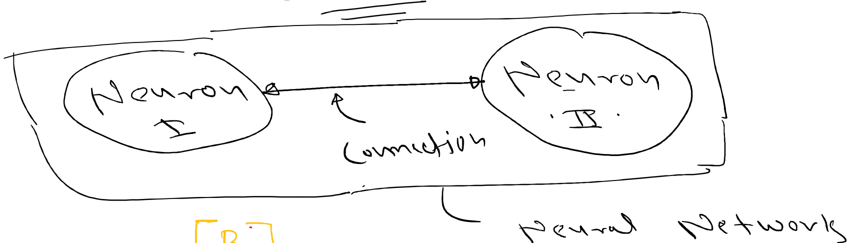
$$p(x \in y=0) = p(0) = 1 - p(1)$$
$$= 0.3$$

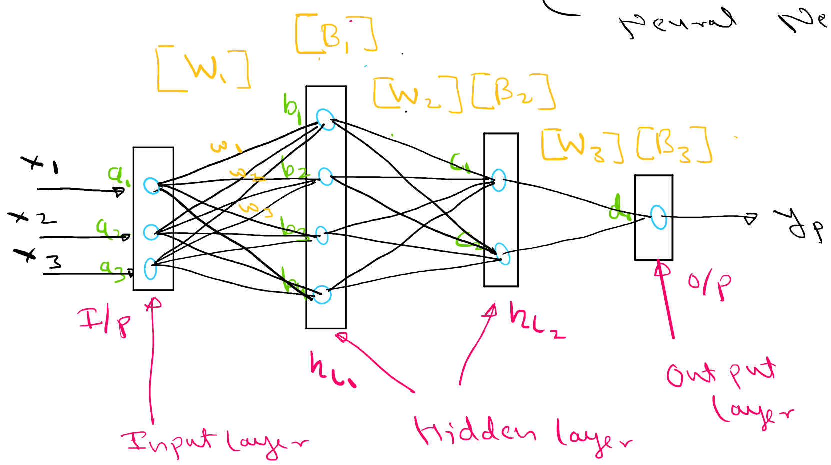
$\frac{1}{0.7}$ $\quad$ $\frac{0}{0.3}$

threshold $\rightarrow 0.5$

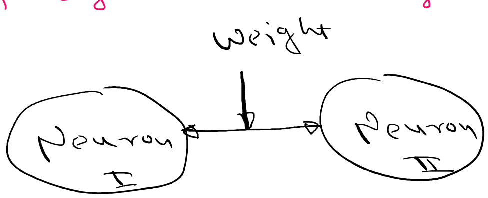
* Structure of Neural Network :-



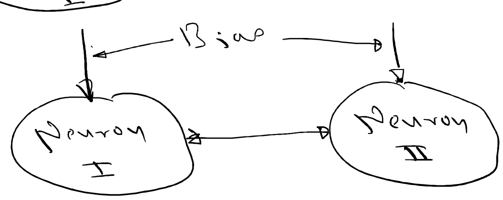




Weight



Bias



$$[W_1] = [w_1, w_2, w_3, w_4 ------ w_{12}]$$

$$[B_1] = [b_1, b_2, b_3, b_4]$$

* Input layer → It hold the input and feed them to hidden layer.

* Hidden layer → Done all the processing/learning

* Out put layer → It shows the out put

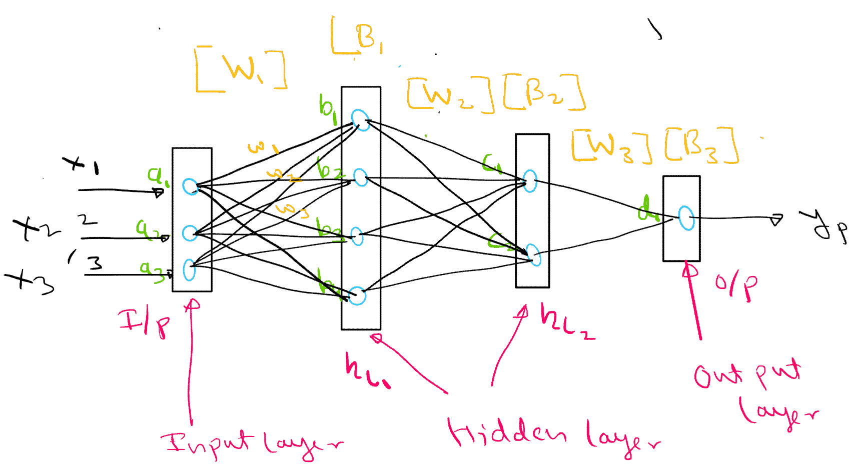
* Input layer → Should be one
* Hidden layer → One or more than one

  One → Shallow Neural Network

  More than one → Deep Neural Network

* Out put layer → Should be one

* How data flow in Neural Network



* No of Neuron in Input layer = No of input features

* No of Neuron in out put layer = decided by the activation funⁿ and No of event.

Case I → let say we are dealing with binary classification and in our out put layer

Case I → let say we are dealing with binary Classification and in our out put layer we have Sigmoid activation funn

No of Neuron in out put layer = One.

* Case II → let's say we are dealing with multi class classification and in our out put layer we have Soft max as a activation function. and No. of ercut is 5.

No of Neuron in out put layer = 5

* Case III → let's say we are dealing with regression problem and in our out put layer we are relu as a activation.

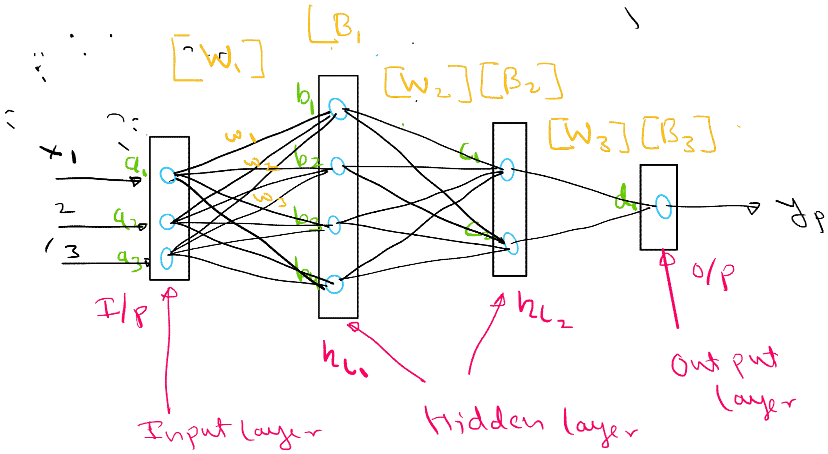
No of Neuron in out put layer = One

* No of Neuron in Hidden layer = Not fixed

General thumb rule (Manually build → ANN)

15 Neuron Start → input layer → Hidden layer → 3 Neuron end out put layer

Hidden layer ↓ in between 3 to 15



Shape of Input layer = (1, 3)
(4, 3).
Stage I

Shape of hidden layer 1 = (1, 4)
(2, 4)
Stage II

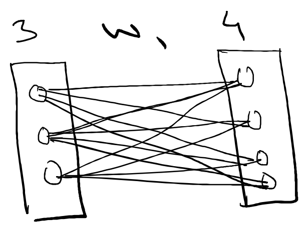
Shape of Hidden layer 2 = (1, 2)
(1, 2)
Stage III

Shape of out put layer = (1, 1)

→ How to control Shape of layers in Neural Networks

→ let's Consider w_1 (weight matrix)

we start from 3 Neuron and we end with 4 Neuron



$w_1 \rightarrow$ 12 weights.

Shape of w_1 = (4, 3)

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

Start $\rightarrow$ (1, 3) (4, 3)

end $\rightarrow$ (1, 4)

$(1,3) \cdot (4,3) = (1,4)$

Transpose

$(1,3)(3,4) = (1,4)$

$(1,4) = (1,4)$

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

* Importance of Bias

Bias $\rightarrow$ One dimensional array that consist values of Bias.

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

	x_1	x_2	x_3	y
1	100	0	300	cat
2	200	3	400	Dog

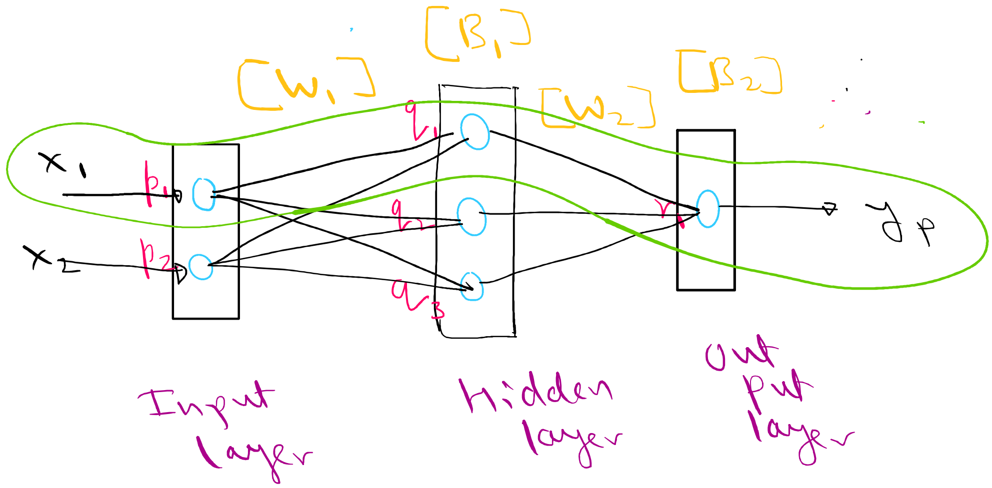
$$z = w_1x_1 + b_1 + w_2x_2 + b_2 + w_3x_3 + b_3$$

$$= 100w_1 + b_1 + 0 + b_2 + 300w_3 + b_3$$

* How Neural Network learn

Chain Rule

$[B_1]$



(1) $x_1 \to p_1 \to q_1 \to r_1 \to y_{p_1}$

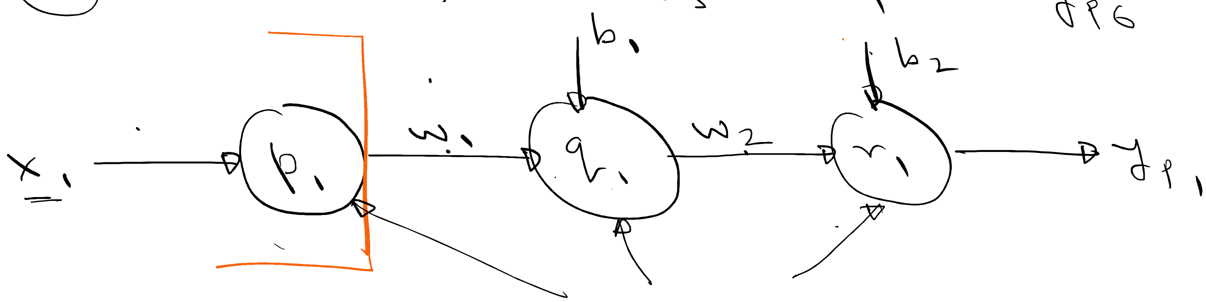
(2) $x_1 \to p_1 \to q_2 \to r_1 \to y_{p_2}$

(3) $x_1 \to p_1 \to q_3 \to r_1 \to y_{p_3}$

(4) $x_2 \to p_2 \to q_1 \to r_1 \to y_{p_4}$

(5) $x_2 \to p_2 \to q_2 \to r_1 \to y_{p_5}$

(6) $x_2 \to p_2 \to q_3 \to r_1 \to y_{p_6}$



Perceptron (Neuron)

$x_1 \to q_1$

Stage I $\to z_1 = w_1 x_1 + b_1$

Stage II $\to \sigma(z_1) = \frac{1}{1+e^{-z_1}} = a_1$

$a_1 \to r_1$

Stage I $\to z_2 = w_2 a_1 + b_2$

Stage II $\to \sigma(z_2) = \frac{1}{1+e^{-z_2}} = y_{p_1}$

stage 4 $\quad 1+e^{-z_2}$

$y_a \quad y_{p_1}$

$\text{loss}(L_1) = y_a - y_{p_1}$

Linear Regression

$y = mx + c$

Best value of m and c

Gradient Decent algorithm

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

$c_{new} = c_{old} - \alpha \frac{\partial L}{\partial c}$

ANN

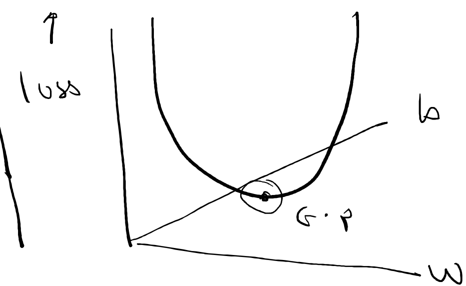
$z = \sum_{i=1}^{n} w_i x_i + b$

Best value of w and b

Gradient Decent algorithm.

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

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



$w_1' = w_1 - \alpha \frac{\partial L_1}{\partial w_1}$

$b_1' = b_1 - \alpha \frac{\partial L_1}{\partial b_1}$

$x_1 \rightarrow p_1 \xrightarrow{w_1'} a_2 \xrightarrow{w_2'} r_1 \rightarrow y_{p_2}, L_2$ $\quad (b_1' \text{ on } a_2, \; b_2' \text{ on } r_1)$

$\min[L_1, L_2, L_3, L_4, L_5, L_6]$

min [L_1 , L_2 , L_3 , L_4 , L_5 , L_6 .]

minimum loss

→ Best value of W and b

→ Best Summation funn

→ activation funn

→ final Op.