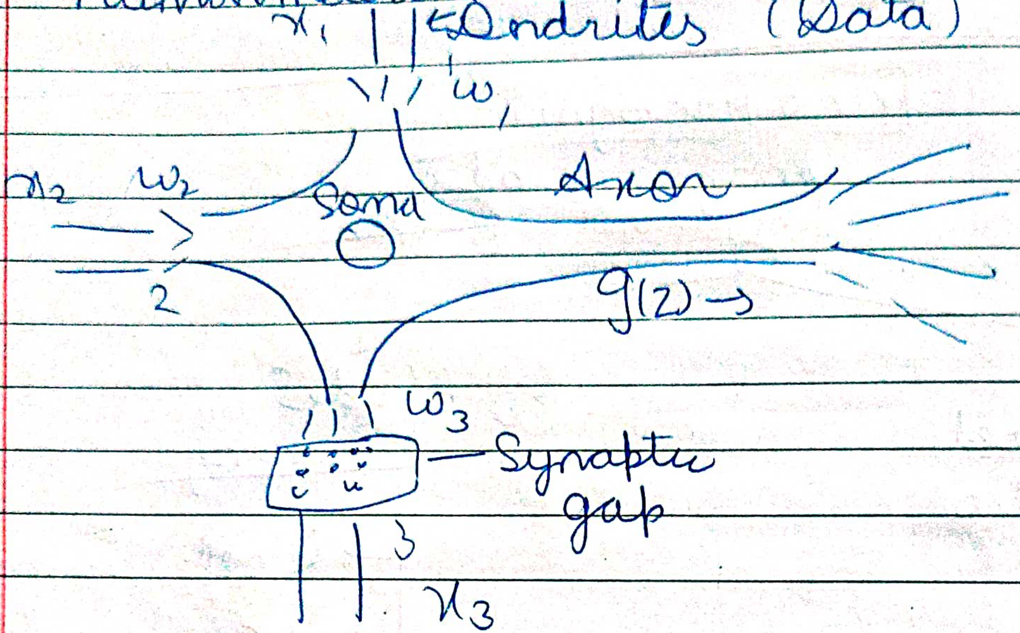
Neural Networks

→ Perceptron
→ Similarity to human neuron

Human neuron

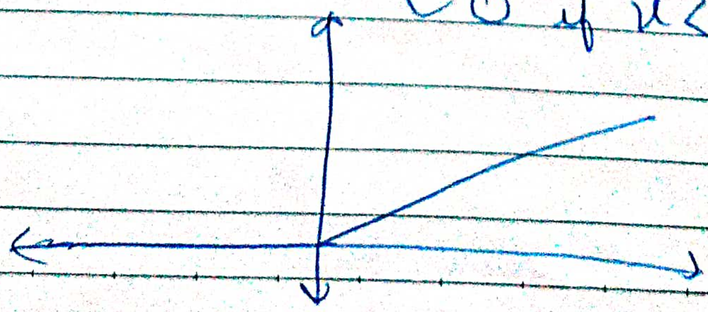


$$z = w_1 x_1 + w_2 x_2 + w_3 x_3$$

w → weight
x – data.

$g(z) \rightarrow$ activation function
↳ can be sigmoid
RELU
↳ rectified linear unit

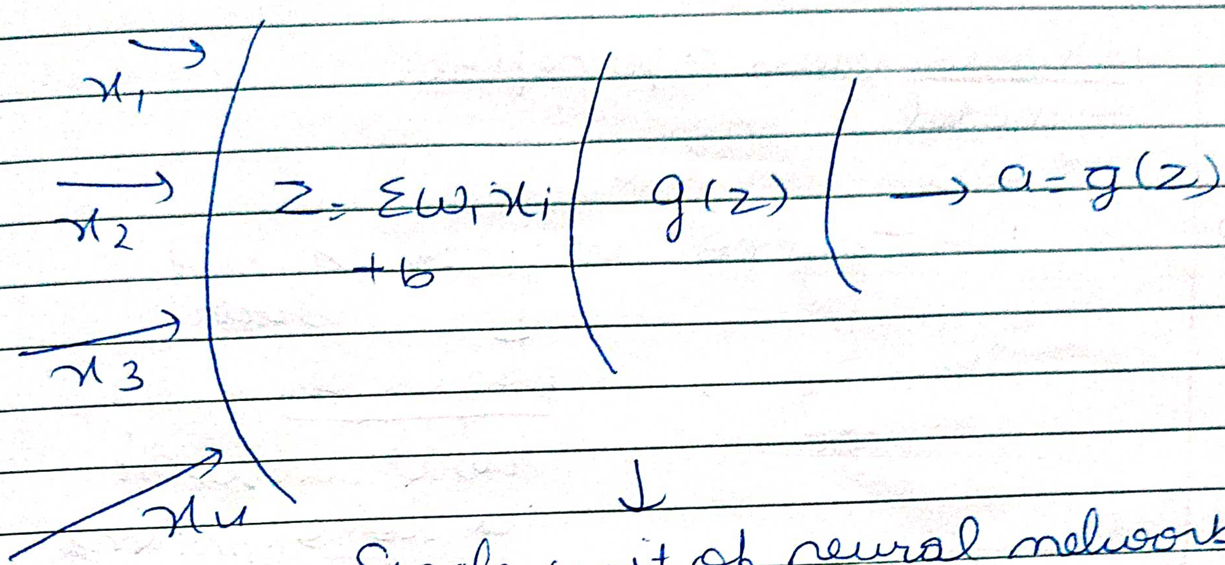
$$\text{ReLU} \rightarrow f(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$



$Z = \sum w_i x_i + b$

$\hookrightarrow$ bias (decides when a neuron fires)

Artificial neuron



↓

Single unit of neural network

↓

Perceptron

Neuron	Perceptron
→ Dendrites	→ Input
→ Synaptic gap	→ weight
→ Soma	→ $g(z)$
→ Axon	→ output.

DATE : / /
PAGE :

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

↓

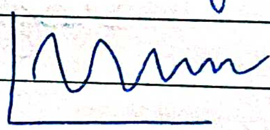
Perceptron

$0 \to 1$

↳ we uses gradient descent
↳ can act as classifier

How to train a perceptron

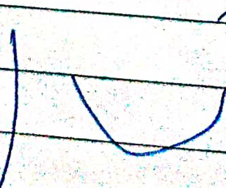
→ model
→ loss
↳ can not use mean squared error
↓
non convex
↓
as it consists of a lot of minima



So we use
→ Log loss / Binary Cross Entropy
↓

$$J(w) = -\sum_{i=1}^{m} \left(y^i \log \hat{y}(i) + (1-y(i)) \log (1-\hat{y}^{(i)}\right)$$

↓
Convex function
↓
Only one global minima



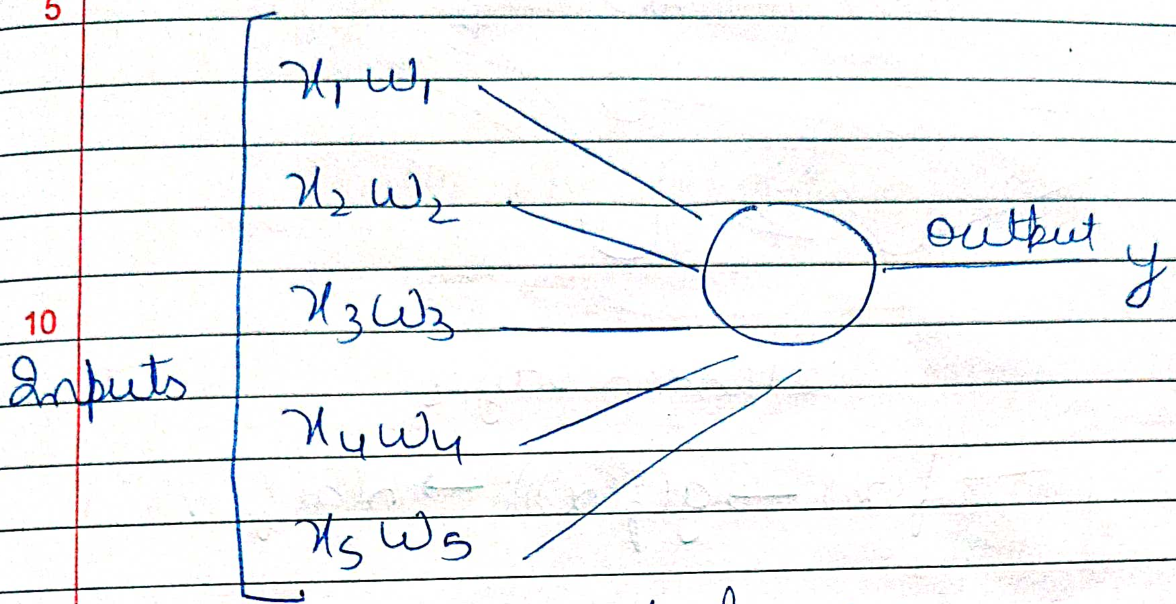
Updated weight

$$w = w - \eta \frac{\partial J}{\partial w}$$

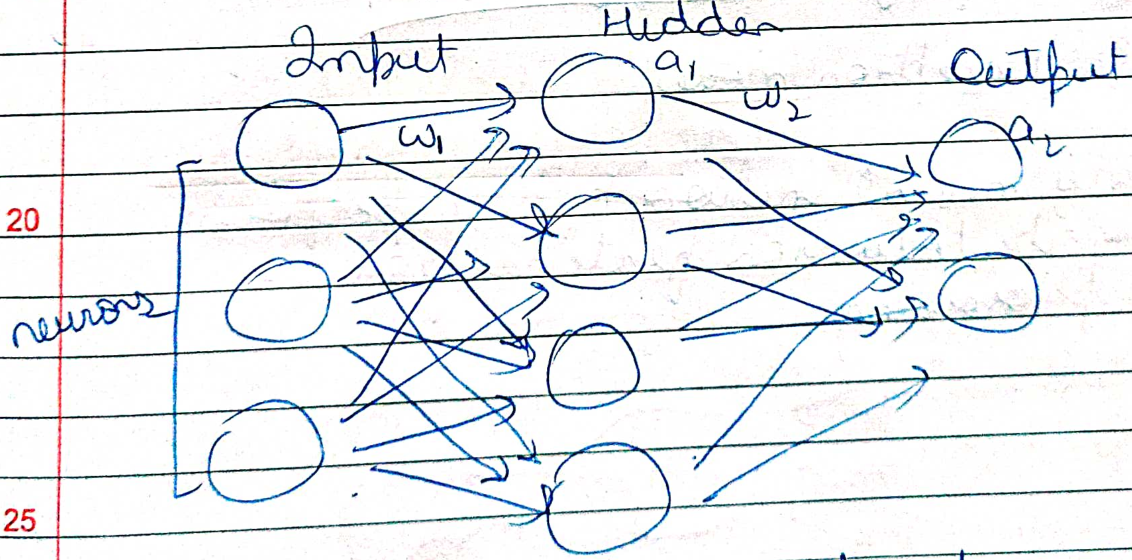
DATE : / / PAGE :

ANN

Perceptron



1-layer

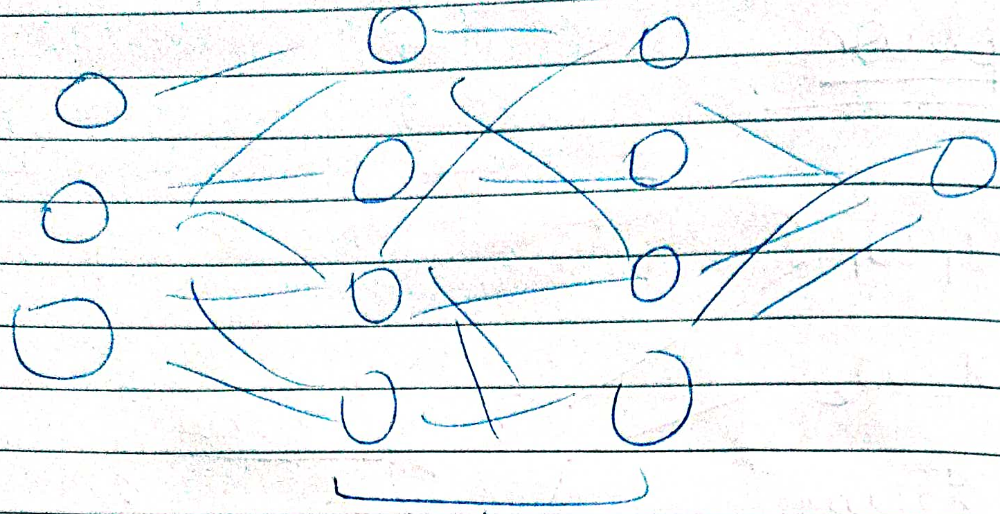


2-layer network

$$\begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix}_{3\times 4} \rightarrow \begin{bmatrix} w_1^{[1]} & w_2^{[1]} & w_3^{[1]} & w_4^{[1]} \end{bmatrix}$$

$w_n^{[l]} \rightarrow l^{th}$ Hidden layer, n^{th} neuron

3-layer neural network



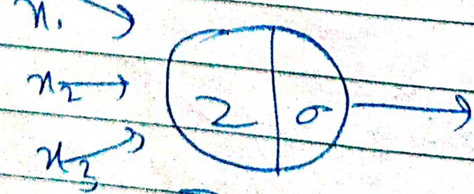
2 Hidden layer

$x \longrightarrow f(x) \longrightarrow g(f(x)) \longrightarrow h(g(f(x)))$

Why multilayer Perceptron

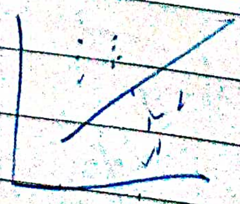
* Single Perceptron model

any neural network that perform computation on features you are providing



$z = wx + b$

works for linearly separable data



eg AND

	T	F
T	T	F
F	F	F

→ linearly sepberable

For multi seperable data , we need multi layer perceptron

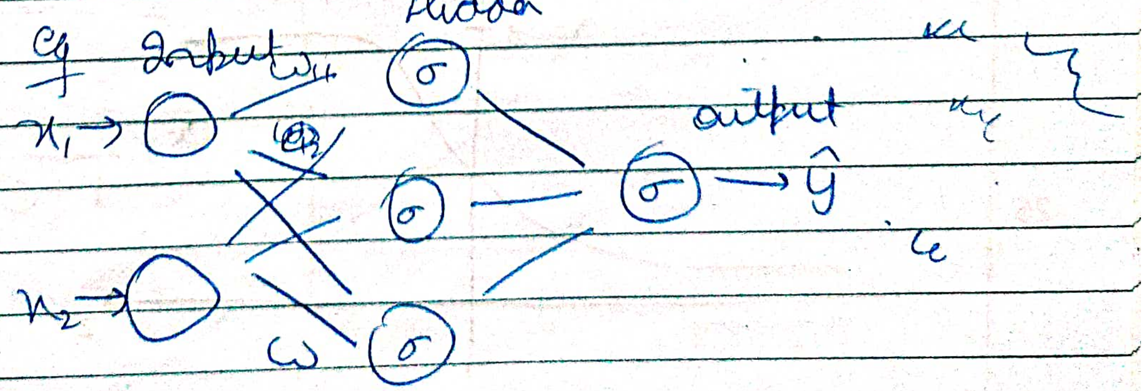
eg ~~XOR~~

* [!AND] AND [OR] → XOR

Architecture

* Input layer
* Hidden layers (n)
* Output layer

eg



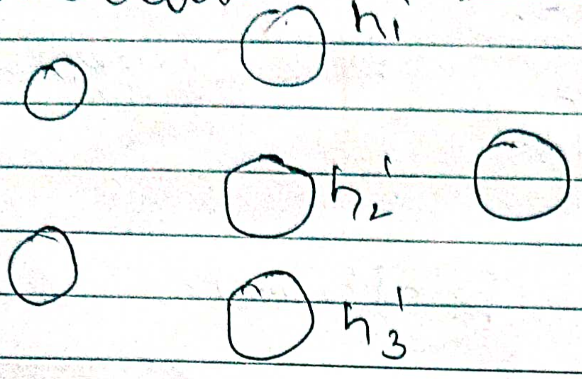
No. of rows − m

No. of columns → n

DATE : __/__/__
PAGE : ____

~~3~~ 2-steps

① Forward Propogation → Backword



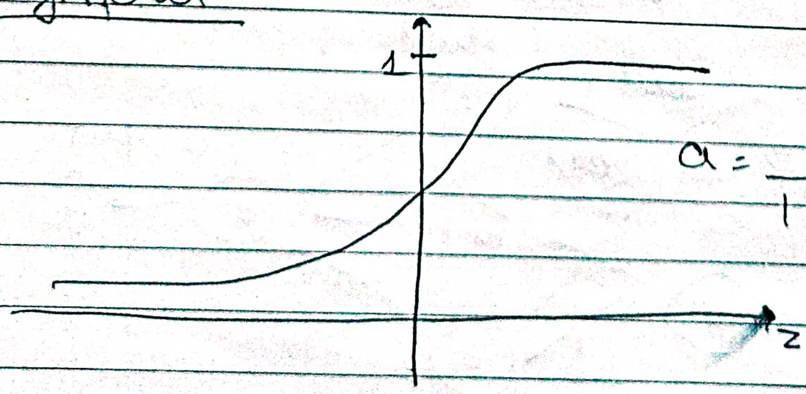
$$h_1^{[1]} = w_1^{(1)} x + b_1^{[1]} \quad , \quad a_1^{[1]} = \sigma(h_1^{(1)})$$

$$h_2^{[1]} = w_2^{[1]} x + b_2^{[1]} \quad ; \quad a_2^{(1)} = \sigma(h_2^{[1]})$$

$$h_3^{[1]} = w_3^{[1]} x + b_3^{[1]} \quad a_3^{(1)} = \sigma(h_3^{[1]})$$

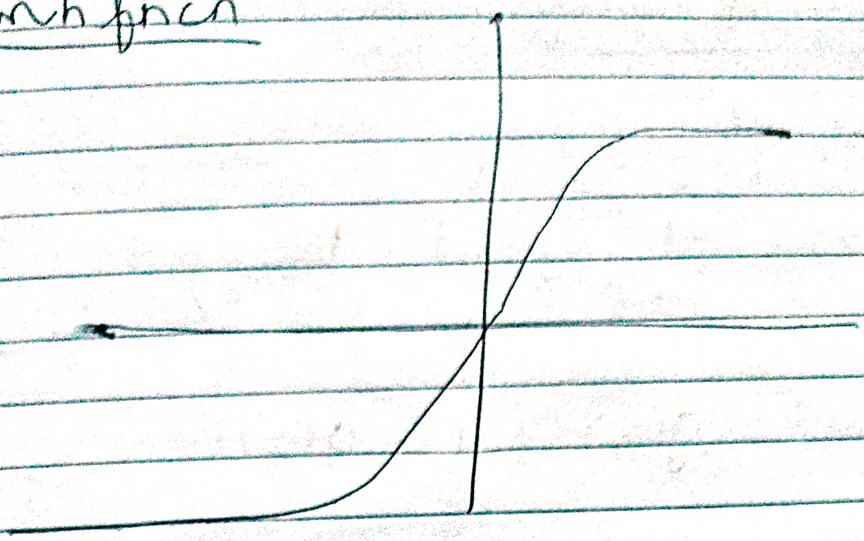
Activation functions

* Sigmoid



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

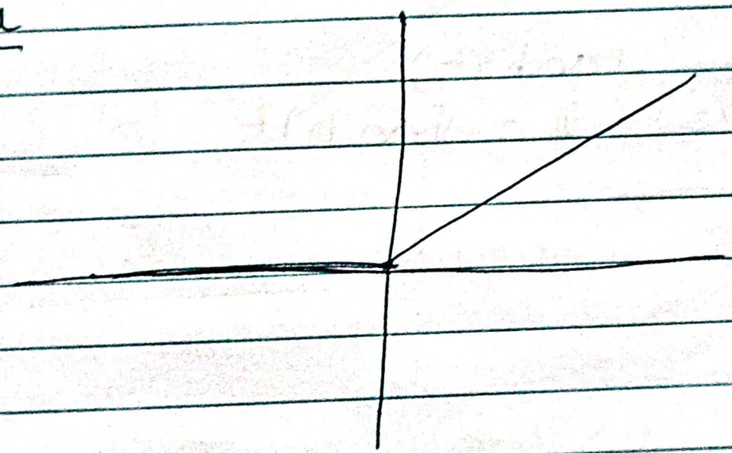
Tanh fncn



$a = \text{Tanh}(z)$

$= \dfrac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$

Relu

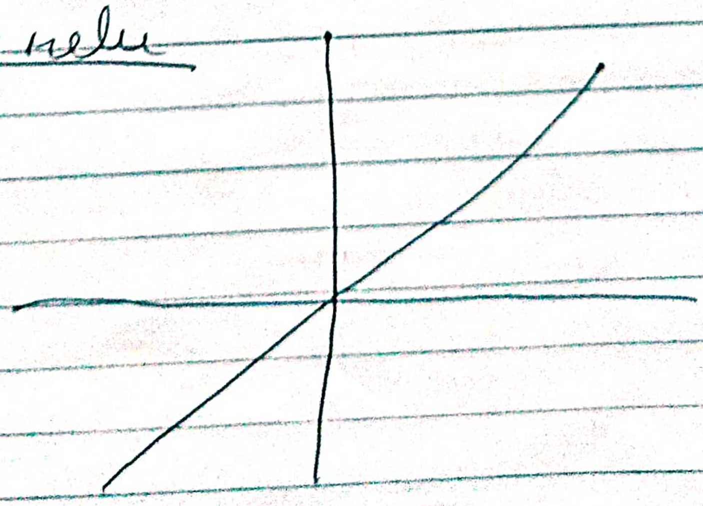


$a = \max(0, z)$

0 if $z = -ve$

z if $z = +ve$

leaky relu



derivation of sigmoid

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

$$g'(z) = \frac{1}{1+e^{-z}}\left(1 - \frac{1}{1+e^{-z}}\right)$$

$$= g(z)\left(1 - g(z)\right)$$

Derivation of Tanh(z)

$$g(z) = \tanh(z)$$

$$g'(z) = 1 - (\tanh)^2$$