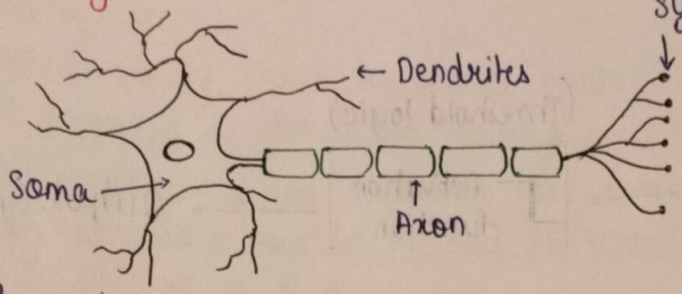
UNIT 6 - INTRODUCTION TO ANN

Perception Learning -

Biological Neuron



Neuron

It is a fundamental unit of nervous system, responsible for receiving, processing & transmitting through electrical & chemical signals.

It mainly has 4 parts -

Dendrites - receives signals from other neurons.

Soma - processes the information

Axon - transmits the output of this neuron.

Synapse - point of connection to other neurons / nerve ending.

A neuron takes an input signal (Dendrites), processes it like the CPU (Soma), and passes the output through a cable like structure (Axon) to other connected neurons (synapse to other neuron's dendrites).

ANN (Artificial Neural Network)

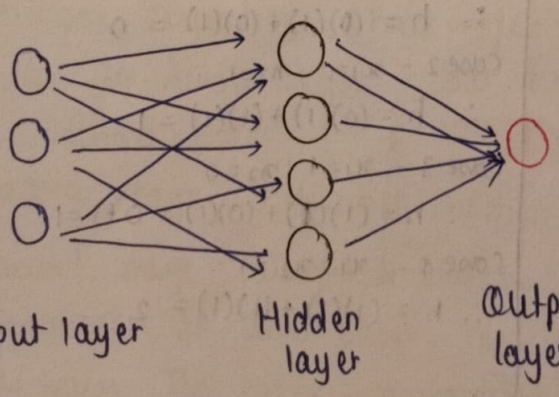
↳ Mimics a network of neurons that makes up a human brain.

↳ Neural Network is inspired by working of the human brain. This in computer science terms is called ANN.

↳ ANN are comprised of node layers, containing : ① input layers ② one / more hidden layers and ③ an output layer

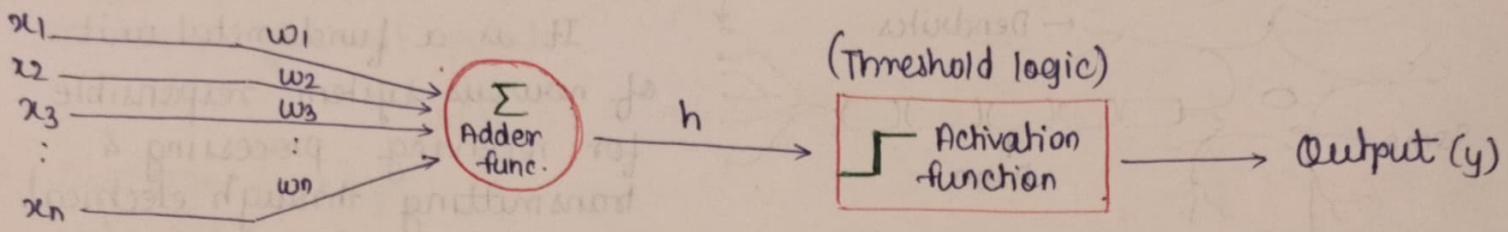
↳ Neural Networks rely on training data to learn & improve their accuracy.

↳ Receives the raw data (Input layer) → Processes the input data by transforming it through weighted connections & activation function (Hidden layers) → Produces final results (Output layer).



McCulloh Pitts Neuron

Very first ANN mathematical model, represents biological neuron.
Proposed by Warren McCulloch & Walter Pitts in 1943.



Explaination:

Inputs - These are signals received by the neuron. Each input can be either excitatory (neuron fires) or inhibitory (not fire).
Here, it is denoted by $x_1, x_2 .. x_n$.

Weights - Each input has equal weight assigned to it of 1.
It is denoted by $w_1, w_2 .. w_n$.

Weighted sum Calculation - Neuron sums the weighted inputs. And outputs as h

$$\therefore \quad h = \sum_{i=0}^{n} (x_i w_i)$$

Threshold - Every neuron has threshold value that determines whether it will fire / not.
(θ)

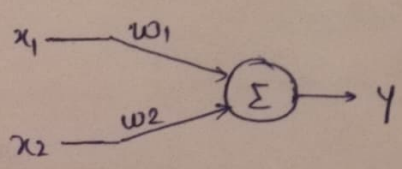
If $h \geq \theta$, then neuron fires & output = 1
$h < \theta$, then neuron doesnot fire & output = 0.

Output - The output is binary; if neuron fires o/p = 1 else o/p = 0.
Denoted by 'y'.

The MCP model can represent basic logic gates (AND, OR, NOT) by appropriately setting the weights and thresholds.

AND Gate

Outputs 1 if all inputs are 1
Assuming w_1 & $w_2 = 1$



x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

Case1 - $x_1 = x_2 = 0$
$\therefore h = (0)(1) + (0)(1) = 0$

Case 2 - $x_1 = 0$; $x_2 = 1$
$\therefore h = (0)(1) + (1)(1) = 1$

Case 3 - $x_1 = 1$; $x_2 = 0$
$\therefore h = (1)(1) + (0)(1) = 0 + 1 = 1$

Case 4 - $x_1 = x_2 = 1$
$\therefore h = (1)(1) + (1)(1) = 2$

$\theta \geq nw - p$ / directly set $\theta = 1.5$

$\therefore$ $\theta \geq 1.5 \Rightarrow y = 1$

$< 1.5 \Rightarrow y = 0$

Perceptron

↳ Introduced by Frank Rosenblatt in 1957. Building block of ANN

↳ He proposed a Perceptron learning rule based on the original MCP neuron.

↳ A Perceptron is an algorithm for supervised learning of binary classifiers.

↳ This algorithm enables neurons to learn & process elements in the training set one at a time.

↳ There are 2 types:

Single layer → can learn only linearly separable patterns

Multi layer → has 2/more layers having great processing powers

↳ Perceptron learning Rule - allows algo to automatically learn the best weights for input features.

Perception Learning Algorithm

Step 1: Initialise with random weights and bias term for each input feature

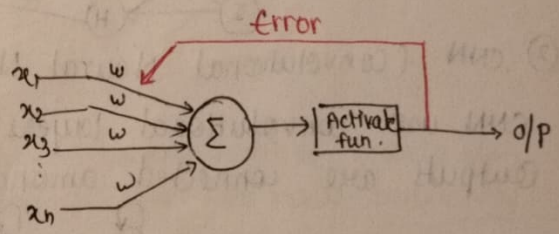
Step 2: Forward Pass

for each input sample, calculate weighted sum

$$n = \Sigma (w_i x_i) + b$$

Apply activation function

$$\text{Output} = \begin{cases} 1 & \text{if } n \geq 0 \\ 0 & \text{otherwise} \end{cases}$$



Step 3: Error Calculation

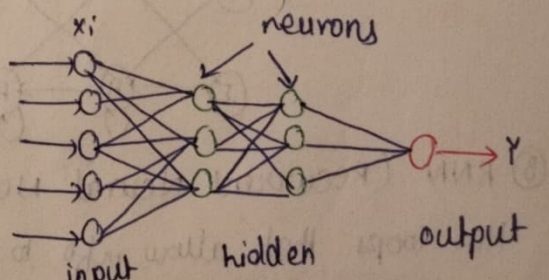
$$E = y_{true} - y_{predicted}$$

Step 4: weight Update

$$w_i = w_i + \eta \cdot (\text{target} - \text{output}) \cdot x_i$$

$$= w_i + \eta\,(\epsilon) \cdot x_i$$

↑ learning rate



Multilayer Perceptron (Backpropogation Algorithm)

Similar to Single layer, but has multiple hidden layers.

Has 2 stages:

forward stage: Activation function starts from i/p layer & terminate on o/p layer

Backward stage: Weight & bias values are modified as per model's requirement. In this stage, the error bet^n actual o/p & target is calculated at o/p layer. The error is propagated backward through network to update weights

MP Neurons ✗ Boolean inputs ✗ Linear ✗ Input are not weighted ✓ Adjustable threshold.

Perceptron ✓ ✗ Linear ✓ diff. for each ✓

Drawback of MLP : Vanishing Gradient Problem

When gradients used in backpropogation becomes very small, effectively "vanishing" as they are propogated back from the output layer to earlier layers. This leads to minimal updates for the weights in the initial layers, making it difficult for network to learn

Causes → Activation func (eg sigmoid / tanh) & Weight Initialization

Consequences → Slow learning & Stalled Learning.

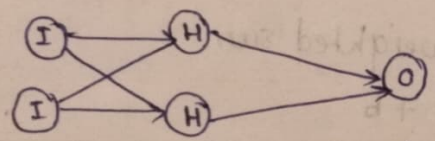
Solutions → ReLU, Weight Initialization techniques, Residual Networks.

NN Architectures

① FNN (Feedforward Neural Networks)

Information flow in 1D – from input to output – without cycles

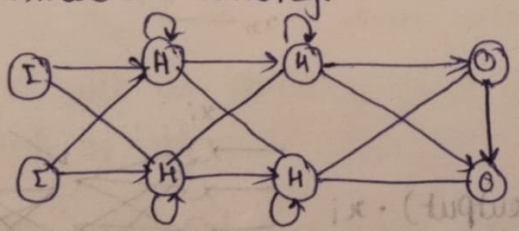
FNN is used for tasks like classification & regression



② CNN (Convolutional Neural Networks)

CNN use convolutional layers to detect patterns in data like edges, textures.

Outputs are connected amongst themselves



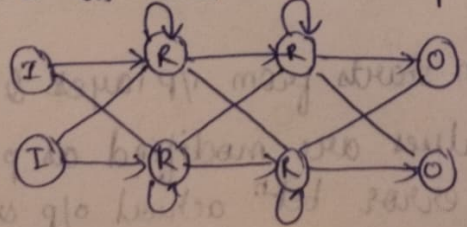
structured grid data like image

③ RNN (Recurrent Neural Networks)

Has loops that allow info to persist, making them suitable for tasks like prediction, NLP, etc.

Input to hidden n/w is delayed in time.

Used where there is need to access previous info in current itteration



effective when context from previous input is used

R → recurrent

Activation functions

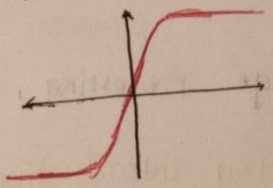
These func are crucial components of NN that introduce non-linearity into the model, allowing it to learn complex patterns.

① Tanh (Hyperbolic Tangent)

Tanh activation func transforms input values to produce outputs betⁿ -1 & 1

Characteristics :-

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



① output ranges from -1 to 1, making it zero centered.

② Non-linearity - allow n/w to learn complex relationships.

③ Gradient - provides stronger gradient during backpropogation.

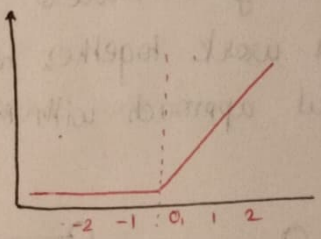
Limitation :-

Vanishing gradient problem.

② ReLU (Rectified Linear Unit)

Characteristics :-

$$ReLU(x) = \max(0, x)$$



① Output range - o/p = 0 for -ve inputs & linear for +ve inputs, ie allowing only +ve values to pass through.

② Simplicity - simple & overcomes vanishing gradient problem

③ Sparsity - more efficient representation.

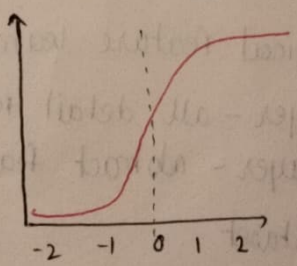
Limitations :-

dying ReLU ~~limitation~~ problem, where neurons can become inactive & only o/p zeros if they receive -ve inputs constantly.

③ Sigmoid (Logistic function)

ranges betⁿ 0 to 1.

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



Characteristics :-

① Output Range - 0 to 1, making it suitable for binary classification tasks.

② Non-linearity

Limitations

① Vanishing gradient problem

② outputs are not zero-centered, which can lead to inefficient weight updates during training.

MLP (contd.)

Advantages -

(i) Versatile - both classification & regression
(ii) Non-linearity
(iii) Parallel computation.

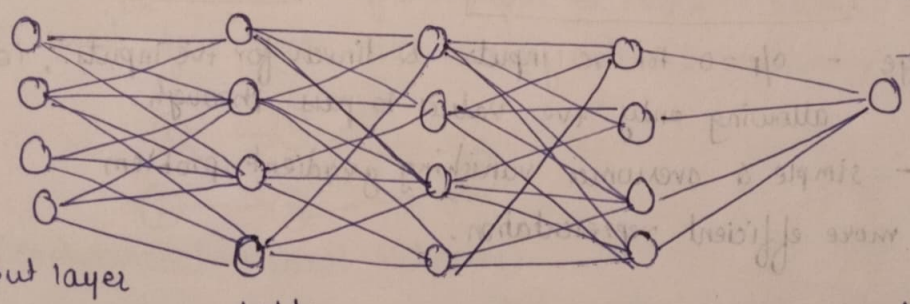
Limitations

(i) Computationally expensive
(ii) Prone to overfitting
(iii) Sensitive to scaling

Appln :-
NLP, Image Processing, Speech Recognition, forecasting

Deep learning Introduction

↳ Deep learning is specilized subset of ML that utilizes ANN with multiple layers to model complex patterns in data.
↳ Inspired by structure & function of human brain, where interconnected neurons work together to process information.
↳ powerful approach with ML to learn complex patterns.



Characteristics :-

(1) Hierarchical feature learning
low layer - all detail features
higher layer - abstract feature knowledge
(2) Large Dataset
(3) Multiple layers - complex feature extraction support
(4) Non-linearity
(5) End to End Learning - learns from raw data
(6) High Computational Requirements - GPU's

Applications :-

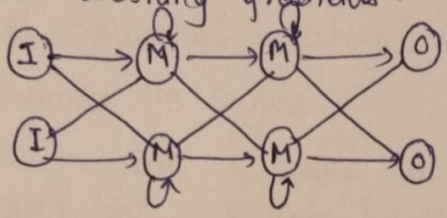
(i) Image Recognition
(ii) NLP
(iii) Speech Recognition
(iv) Self driving cars
(v) Healthcare
(vi) finance
(vii) Recommendation Systems.

Architectures of Deep learning:-

FNN , CNN , RNN

④ LSTM (Long Short Term Memory Networks)

includes memory cells & gates to better retrain info over long sequencing issues like vanishing gradients.



M → memory.

Limitations of NN Architectures.

① Requires large amount of data.
② Requires more computational power & time.
③ Overfitting.
④ Complex in design.
⑤ Hard to understand how actually is their internal working.