Neural Networks & Deep Learning :-

Unit - I

⊕ Artificial Intelligence :-

→ Commonly referred as AI, which is the process of imparting data, information, & human intelligence to machines.

→ It's main goal is to develop self-reliant machines that can think and act like humans.

→ These machines can mimic human behaviour & human intelligence to machines. They can perform tasks by learning & problem solving. Most of AI systems simulate natural intelligence to solve complex problems.

⊕ Machine Learning :-

→ It is a discipline of computer science that uses computer algorithms & analytics to build predictive models that can solve business problems.

→ Types :-

Supervised learning - Labelled data
Unsupervised learning - Unlabelled data
Reinforcement learning - train agent to complete task with uncertain E.

Deep Learning:-

→ It is a subset of ML that deals with algorithms inspired by the structure & function of the human brain.

→ Deep Learning Alg. can work with an enormous amount of both structured & unstructured data.

→ Deep learning's core concepts lies in Artificial Neural Networks, which enable machines to make decisions.

Artificial Neural Networks:-

→ ANNs are algorithms based on brain function and are used to model complicated patterns & forecast issues.

→ It is a Deep Learning method that arose from the concept of the human brain biological neural networks.

→ The development of ANNs ~~are exactly~~ was the result of an attempt to replicate the workings of the human brain.

→ The workings of ANN are extremely similar to those of biological neural networks, although they are not identical.

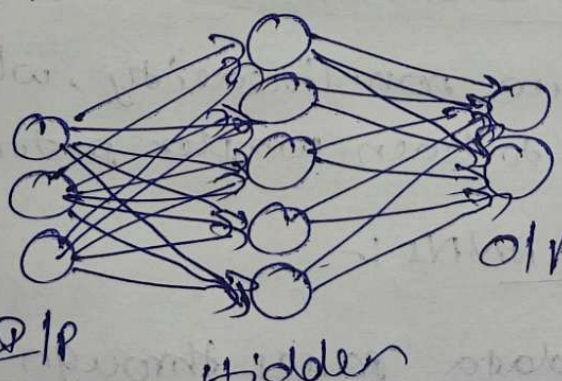
→ ANN algorithms accepts only numeric and structured data.

Structure of ANNs:-

1) Input Layer:
- Recieves i/p data (features)
- No computations are performed here.
- No. of neurons equals the no. of i/p features.

2) Hidden Layers:-
- Processes the data through computations using weights & biases.
- Each neuron in a layer connects to all neurons in next layer.
- Activation functions are applied to use non-linearity, which allows networks to learn complex patterns & relationships.

3) Output Layer:-
- Provides final result of the neural network
- No. of neurons depends on type of problem (classification or regression).



→ Some Activation Functions are:

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

tanh → $\tanh(x)$

ReLU → $\max(0, x)$

Leaky ReLU → $\max(0.1x, x)$

Maxout → $\max(w_1^T x + b_1, w_2^T x + b_2)$

Key components of ANN:-

1) **Neurons**: (Nodes)
Basic units that process data, like biological neurons.

2) **Weights**: These have info & data about i/p signal connection strengths b/w neurons; adjusted during learning.

3) **Bias**:
A value added to the weighted sum of inputs to improve learning & accuracy and shift the o/p to the req.

4) **Activation Functions**:
Introduce non-linearity, which allows networks to learn complex patterns.

→ **Working of ANNs**:-

1) The i/p data passes through the i/p layer.
2) Weights and biases modify the data in the hidden layers.

3) The activation functions are applied to generate o/p.

4) The network learns by updating weights using optimization techniques like Gradient Descent (used to min loss function)

→ During training, the network changes the weights based on training data in the hidden layers to find a solution that min error.

→ Learning Process of ANN:-

The learning process refers to how the network adjusts weights & biases to improve accuracy in predicting o/p based on given i/p.

Steps:-

1) Forward Propogation:-

- Data flows from i/p to o/p layers
- o/p's are calculated.

2) loss Calculation:-

- The difference b/w predicted o/p & actual result.
- loss functions like Mean Squared Error (MSE) or Cross Entropy loss are used.

3) Backward Propogation:-

- The error is propogated backward to adjust weights.
- This reduces loss in next iterations.

→ Applications of ANNs:-

1) Image Recognition: Facial Recognition Systems
2) NLP: Chatbots, Lang. translation
3) Medical Diagnosis: Detection of diseases
4) Fraud Detection: fraud financial transaction
5) Speech Recognition: Virtual Assistants like Alexa

→ Advantages:

1) Can model complex, non-linear relationships
2) Learns & improves over time with data
3) Adaptable to diff. domains.

→ Disadv:-

1) Requires large amounts of data.
2) Computationally expensive
3) Can suffer from overfitting

⊛ Deep Neural Network:-

→ It is an ANN but with multiple hidden layers.

→ All DNNs are ANNs but all ANNs are not DNNs as some of them can have a single hidden layer.

② Some Important Terminologies in ANN:-

→ **Learning Rate:-**

- A hyperparameter that controls the size of weight updates during training.
- Small values lead to slow learning, while large values may cause instability
- Ranges from 0 to 1

→ **Epoch:-**

- One complete pass of entire training dataset through the n/w.

→ **Overfitting:-**

- Occurs when the model performs well on training data but poorly on new data.
- Can be reduced using regularization techniques like dropout.

→ **Weights:-**

- Each neuron is linked to other neurons through connection links that carry weight. The weight has info & data about i/p signal.
- The o/p depends on i/p signal and weights.
- Weights can be presented in a matrix form known as connection matrix (w).

if there are 'n' nodes with 'm' weights

$$w = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & \vdots & & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{nm} \end{bmatrix}$$

→ Bias :-

- Bias is a constant that is added to the product of inputs & weights to calculate the product.
- It is used to shift the result to positive or negative side.
- The net i/p weight is increased by a +ve bias & decreased by a -ve bias.

→ Threshold :-

- A threshold value is a constant value that is compared to the net i/p to the o/p.
- The activation function is defined based on threshold value to calculate the o/p.

⊛ Example :- Self Driving Cars

1) I/p layer :- Car's sensors

Inputs can be images of roads, traffic signals, pedestrians, etc. and speed & distance.

2) Hidden layer

Recognising pedestrians, lanes, estimating best route.

3) O/p layer

Decisions like "apply brakes", "turn left" or "increase speed", etc.

4) weights :-

- If a pedestrian is detected, higher weight is assigned to signaling "apply brakes".
- If road is clear, higher weights is given to "accelerate".

5) Bias :-

- Helps fine tune decision making.
- If road is covered densely by fog, the car may stop because the bias nudges the decision in favour of caution.

6) Activation function

Decides whether to trigger an action or not

- If pedestrian is far, "apply brakes" neuron may remain inactive
- If near, the neuron fires to stop the car.

7) Overfitting

If system remembers only specific situations like one particular intersection, it may fail in new situations.

8) Loss function

Measures how bad car's decision was.

- If car predicted "no need to stop" but almost hit a pedestrian, the loss is high.

9) Backpropogation :-

After a near miss, the system updates to never repeat the mistake.

⊛ Basic Models of ANN :-

1) Single-Layer Perceptron Model :-

→ Contains 1 i/p layer & 1 o/p layer.
→ No hidden layers
→ Each neuron in i/p layer connects directly to the o/p layer.
→ Used for simple linear classification problems.

Working:
→ computes a weighted sum of i/ps, applies activation function & produces an o/p.

Lim:-
→ Cannot solve non-linear problems.

Application:
→ Pattern recognition for simple datasets.

2) Feedforward Neural N/w :-

→ Information flows only in one direction; from i/p to o/p
→ No feedback loops.

working:
→ Processes data layer by layer without going backward.
→ Suitable for supervised learning tasks.

Applications
→ Pattern recognition & predictive modeling.

3) Recurrent Neural Network (RNN) :-

→ Contains feedback connections.
→ O/Ps from neurons are fed back as i/Ps to previous layers.

Working:
→ Maintains memory of previous i/Ps using loops.
→ Suitable for sequential data.

App :-
→ Time-series forecasting, lang. modeling & speech recognition.

4) Convolutional Neural Networks :-

→ Designed to process grid like data such as images.
→ Contains convolutional layers, pooling layers, & fully connected layers

Working :-
→ Extracts features from data using convolution operations.
→ Reduces complexity through pooling while retaining essential features.

App :-
→ Image classification, object detection & video analysis.

⑧ Supervised Learning Networks :-

→ It is a type of ML where the model is trained using labeled data.

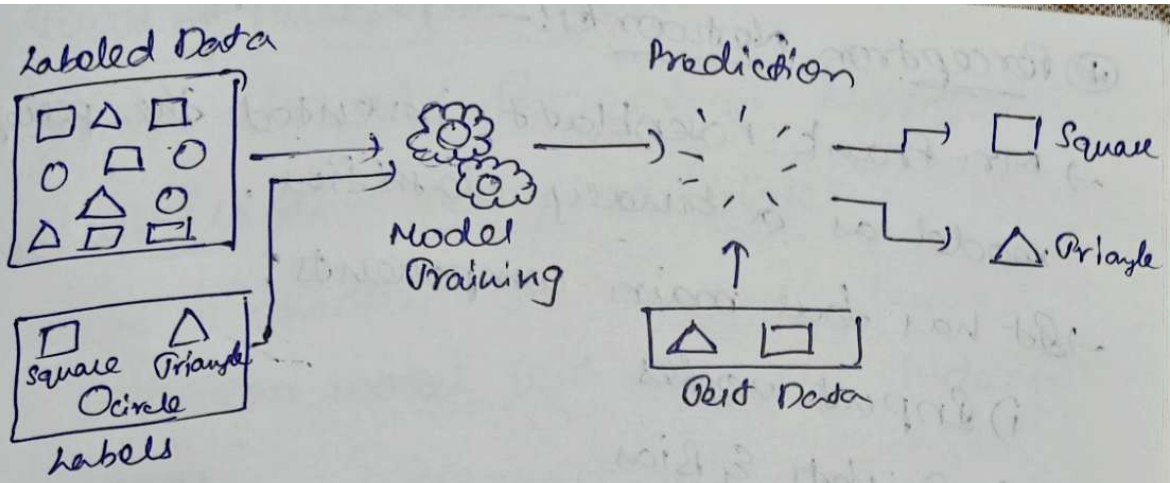
→ The goal is to learn a mapping from i/p data to correct o/p. Here, the model adjusts its parameters (like weights) based on the diff b/w its predicted & actual o/ps.

→ During the training of ANN under supervised learning, the i/p vector is presented to the network, which will produce an o/p vector. This o/p vector is compared with the desired o/p vector. An error is generated if there is a diff and on its basis, the weights are adjusted until actual o/p matches the desired o/p.

→ Some examples/types are ANNs, CNNs, RNNs, Decision trees.

→ Working :-

1) Forward Propogation
2) Loss Calculation
3) Backpropogation
4) Iteration



→ Types:-

1) Regression:-

- Regression algorithms are used if there is a relationship b/w i/p variable & the o/p variable. It is used for the prediction of continuous variables, such as weather forecasting, Market trends, etc.
- Some examples are:
  Linear Regression, Regression Trees, Non-linear Regression, Bayesian Linear Regression.

2) Classification:-

- Classification algorithms are used when the o/p variable is categorical, which means there are two classes such as Yes-No, M-F, True-False, etc.
- Some ex:
  Random Forest, Decision Trees, Logistic Regression, Support Vector Machines.

→ Adv:- High Accuracy, clear objective & variety of applications.

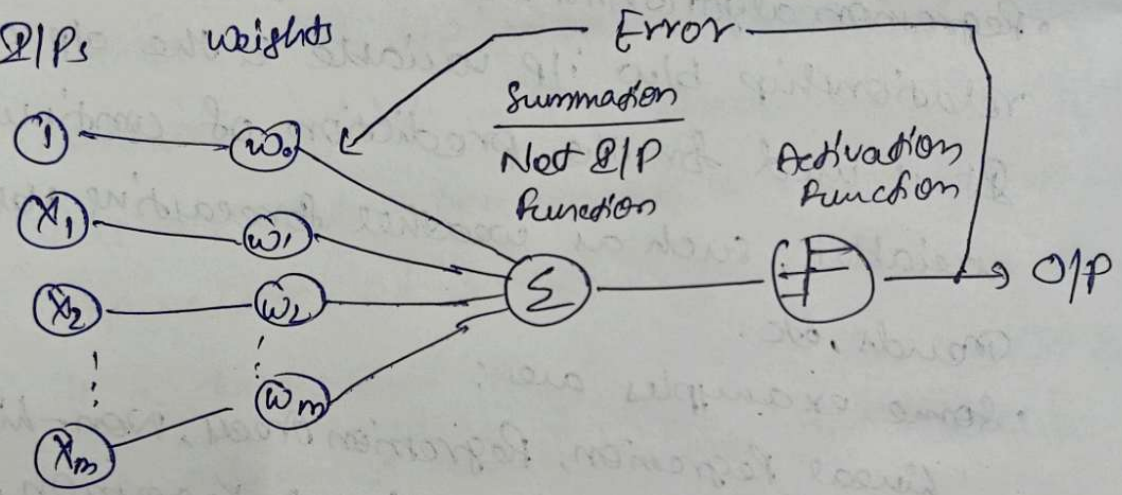
→ Disadv:- Dependence on labeled data, overfitting risk.

④ Perceptron Networks:-

→ Mr. Frank Rosenblatt invented the perceptron model as a binary classifier.

→ It has four main components:

1) Input nodes
2) Weights & Bias
3) Net summation
4) Activation function



→ In the 3rd component, the weighted i/ps are summed up & a bias is added. This sum is then passed to an activation function.

Mathematically,

$$Sum = \sum (w_i x_i) + b$$

where, w_i is weigh,
x_i is i/p
b is bias

→ Activation functions used here are, Step function, Sign function & Sigmoid function.

→ Types of Perceptron Models :-

1) Single-Layered Perceptron Model.

2) Multi-Layered Perceptron Model :-

- It is exactly like a single layered perceptron model. but has more hidden layers.
- Each neuron in a layer connects to all neurons in the next layer (fully connected network).

Working :-

- It uses backpropogation for training by adjusting weights & biases.
- Can solve non-linear & complex problems.

App :-

- Image classification, Speech recognition

Adv :-

- works well with both small & large i/p data.
- helps us to obtain quick predictions after training
- helps to obtain same accuracy ratio with large & small data.

Disadv :-

- Computations are difficult & time consuming.
- Model functioning depends on quality of training.

→ Perceptron Function :-

Perceptron function "$f(x)$" can be achieved as o/p by multiplying the i/p 'x' with learned weigh coefficient 'w'.

Mathematically:

$f(x) = 1$; if $w \cdot x + b > 0$.

otherwise, $f(x) = 0$

(iv) Adaptive Linear Neuron :- (ADALINE).

→ It is a type of artificial neuron used in neural networks, primarily for regression tasks.

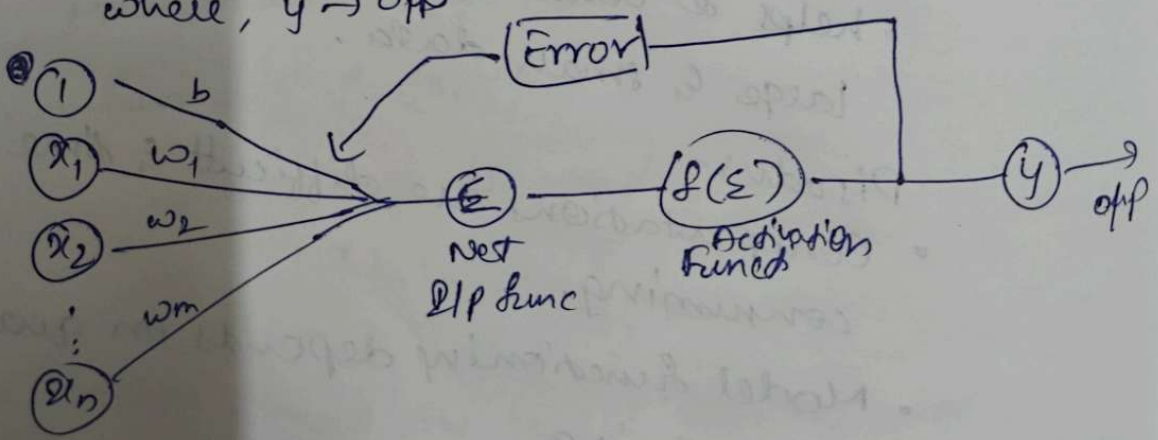
→ It is similar to perceptron but with a key diff. in how it updates weights during training.

→ It uses a linear activation function, meaning the o/p is a weighted sum of i/ps.

→ For each i/p $x_1, x_2, \ldots x_n$, neuron computes a weighted sum :

$y = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n$

where, $y \rightarrow$ o/p



→ First calculate the net i/p to your Adaline n/w then apply activation function to its o/p and compare it with original o/p. If both are equal, then give the o/p or else send back the error and update according to the error.

→ The Adaline uses Mean Square Error to calculate error:

$$E = \frac{1}{2} \sum (y_{actual} - y_{predicted})^2$$

→ Update the weights & bias according to the error and for it use the formula

$$w_i = w_i + \alpha \cdot (y_{actual} - y_{predicted}) \cdot x_i$$

where α is learning rate based on gradient descent method.

→ A small learning rate ensures slow & steady convergence while a large one may cause the model to overshoot.

→ ADALINE is used in tasks ~~with~~ req. linear regression when the relationship b/w i/p & o/p is linear and als in classification when o/p is continuous.

→ Adv:-
It is simple & works well.
It has relatively straightforward training.

→ Disadv:-
limited to linear problems and can't handle non-linear data effectively.

(4) Back-propogation Network:-

→ It is a type of ANN that uses the backpropogation algorithm for training the network.

→ It is widely used in supervised learning, where the network learns from i/p-o/p pairs.

→ Its core purpose is to minimize error in the network by adjusting the weights.

→ It fine tunes the weights of a neural network based on the error rate obtained in the previous epoch (iteration), which makes the model more reliable.

→ The backpropogation process involves two main steps:

1) Forward Pass:

- I/p data is passed through the n/w.
- Each neuron performs a weighted sum of its i/ps, and applies an activation function (sigmoid, tanh or ReLU) and passes the result to the next layer.
- The network produces an o/p for final layer.

2) Backward Pass:-

- After obtaining the o/p, error is calculated by comparing predicted o/p with actual o/p (using error function like MSE)
- This error is propogated backward from the o/p layer towards the i/p layer.

• During the backward pass, the gradient of the error with respect to each weight is computed using gradient descent.

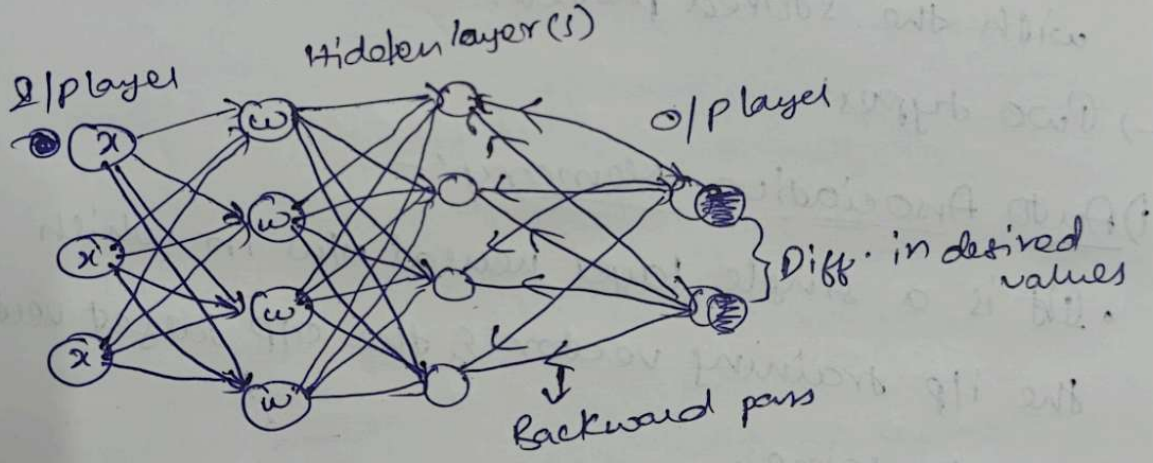
[Gradient Descent: It is a technique used to min the cost function of a model by iteratively adjusting the model parameters to reduce the diff btw predicted & actual values]

• weight updates are the performed using:

$$w = w - \alpha \times \frac{\partial E}{\partial w}$$

$\alpha \rightarrow$ learning rate

$\frac{\partial E}{\partial w} \rightarrow$ derivative of error wrt weight.



→ Adv:-

- Efficient for large n/ws
- Flexible for diff types of n/ws
- Versatile for various problems like classification, regression, pattern recognition

→ Disadv:-

- Computationally expensive
- Vanishing Gradient Problem, where the gradients updating the n/w can become very small or "vanish" leading to slow learning.

(4) Associative Memory Networks:-

→ These kinds of neural networks work on the basis of pattern associations, which means they can store patterns.

→ At time of giving an o/p, they can produce one of the stored pattern by matching them with the given i/p pattern.

→ These types of memories are also called Content-Addressable Memory CAM.

→ Associative Memory makes a parallel search with the stored patterns as data files.

→ Two types:

1) Auto Associative Memory:-

- It is a single layer neural n/w in which the i/p training vector & the o/p target vectors are the same.
- If a partial or noisy version of a pattern is presented, the network retrieves original complete pattern.
- The n/w learns by forming associations b/w similar i/p & o/p patterns. Once trained, it can recognize distorted patterns & correct them.

- Input is associated with itself ($x \rightarrow x$)
- Hopfield Network is a common example.
- App:-
  Pattern completion, Noise filtering, error correction.

2) Hetero-Associative Memory:-

- It stores and recalls patterns where the i/ps & o/ps are different.
- It learns to associate one set of patterns with a different set.
- This network maps i/p patterns to different o/p patterns ($x \rightarrow y$).
- When presented with a known i/p it produces the associated o/p.
- The association can be b/w arbitrary sets of patterns.
- Useful for learning associations b/w two diff sets of patterns.
- Apps:-
  Translation tasks (text to phoneme), signal processing, data mapping.

(4) Training Algorithms for Pattern Association:-

→ Pattern association in neural n/ws refers to the ability to store & recall patterns based on input.

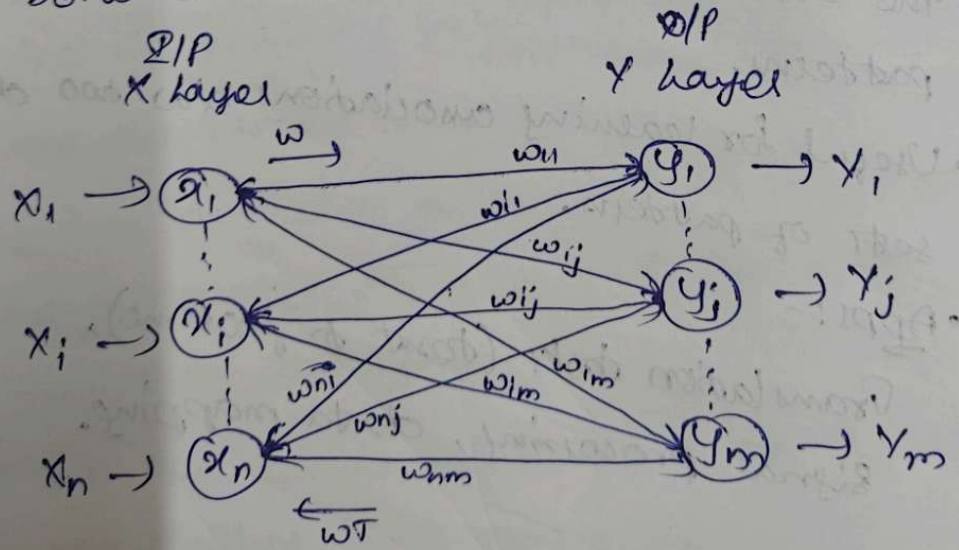
→ When an input pattern is presented, the n/w retrieves the associated o/p pattern.

→ Two key models used are Bidirectional Associative Memory (BAM) & Hopfield Networks.

(a) Bidirectional Associative Memory (BAM):-

→ BAM is a type of RNN designed to associate two related sets of patterns i.e i/p & o/p pattern pairs.

→ It operates in both directions back and forth allowing pattern recall either forward or backward.



→ Every neuron in layer X (i/p layer) is connected to every neuron in layer Y (o/p layer) and its weights are stored in weight matrix W of size n×m.

→ Training Alg:-

1) Initialize weight matrix:

Set all elements of weight matrix W to zero initially.

2) Apply Hebbian learning rule:

For each pattern pair (x, y) where,

x → i/p pattern & y → corresponding o/p

$$w_{ij} = w_{ij} + x_i \cdot y_j$$

where,

w_{ij} → weight connecting neuron i in X to j in Y.

x_i → ith element of i/p layer pattern's

y_j → jth element of o/p layer pattern

3) Normalize weights:

If weights grow too large due to multiple pattern updates, normalize them:

$$w_{ij} = \frac{w_{ij}}{k}$$

k → total no. of pattern pairs

4) Stop condition:-

Stop updating weights when all pattern pairs have been processed.

→ Recall Process:-

1) Forward Recall:-

Provide a pattern 'x' to layer X and compute the o/p pattern in layer Y.

$$Y' = \text{sign}(X \times W)$$

2) Backward Recall :-

Provide a pattern 'Y' to layer Y & compute the associated i/p pattern in layer X:

$$X' = \text{sign}(Y \times W^T)$$

$W^T \rightarrow$ transpose of weigh matrix

3) Reiterate until patterns stabilize

→ It is a type of Hetero-Associate Memory.

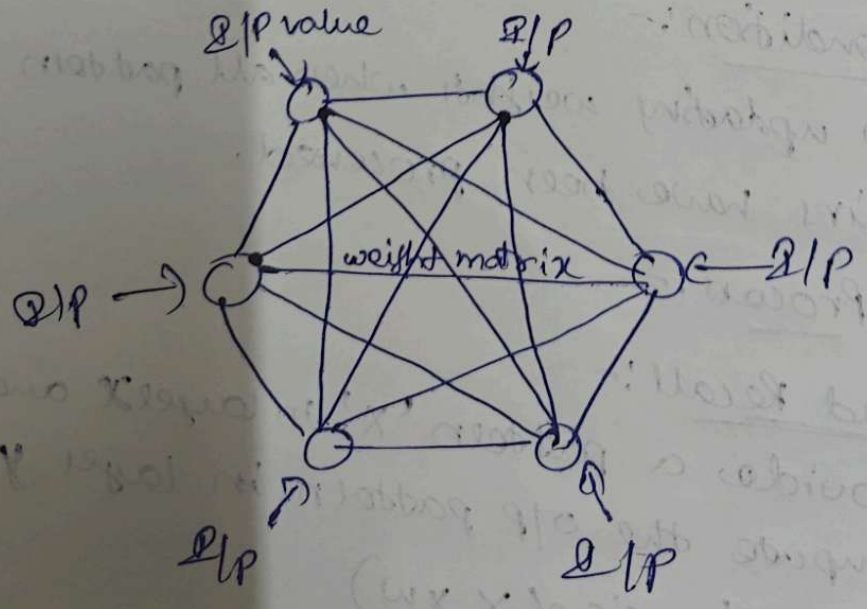
④ Hopfield Network :-

→ It is a fully connected RNN that stores multiple patterns as stable states.

→ It is a form of associative memory, which can recall patterns even when provided with noisy or partial inputs.

→ It has a single layer of fully connected neurons, where each neuron is connected to every other neuron except itself.

→ The connections are represented by a symmetric weigh matrix W of n×n where the weights satisfy the conditions $w_{ij} = w_{ji}$ & $w_{ii} = 0$



→ Training Alg :-

1) Initialize weights :-

Set all diagonal to zero & other weights to zero.

$$w_{ii} = 0$$

2) Hebbian Learning Rule :-

For each training pattern P^k, where P^k is a vector of binary values (+1 or -1) :

$$w_{ij} = w_{ij} + P_i^k \cdot P_j^k$$

where,

w_{ij} → weights b/w neurons i & j

P_i^k → value of ith element of P^k

3) Ensure Symmetry of weights :

$$w_{ij} = w_{ji}$$

4) Normalize weights :-

Divide each weight by the no. of patterns

$$w_{ij} = \frac{w_{ij}}{k}$$

→ Recall :-

1) Provide input by presenting a partial or noisy i/p pattern.

2) Update Neurons :-

For each neuron, update its state based on weighted inputs :

$$v_i = \text{sign}\left(\sum_{j=1} w_{ij} v_j\right)$$

where,

v_i ⇒ new state of neuron

the sign function returns +1 if sum is +ve and -1 if -ve.

3) Repeat until the n/w stabilizes or converges to a stored pattern.

→ Adv :-

- Functions well with noisy or incomplete inputs.
- Simple structure & robust convergence properties.