NNDL

unit-1

ANN

it is a computational models inspired by the human brain's neural networks. They are a subset of machine learning and form the foundation of deep learning.

① **Neurons (Nodes):-**

The basic buildins blocks of ANNs. Inspired by biological neurons.

② **layers:-**

input, hidden, output

③ **weights and Biases**

weights → parameters that determine the strength of the connection between neurons.

Biases → Additional parameters that allow the model to fit the data better.

4. Activation function

→ Sigmoid → maps input to a value between 0 and 1

→ ReLU (Rectified Linear Unit) → outputs the input directly if positive; otherwise, output 0.

→ Tanh : maps input to a value between -1 and 1

5. forward propagation :-

→ The process of passing input data through the network to compute the output

→ Each layer applies weights, biases and activation function to the input

6. Back Propagation

→ The process of updating weights and biases to minimize the loss

→ used gradient descent to adjust parameter based on the error.

Epochs and Batch size

Epoch → one complete pass through the entire training dataset.

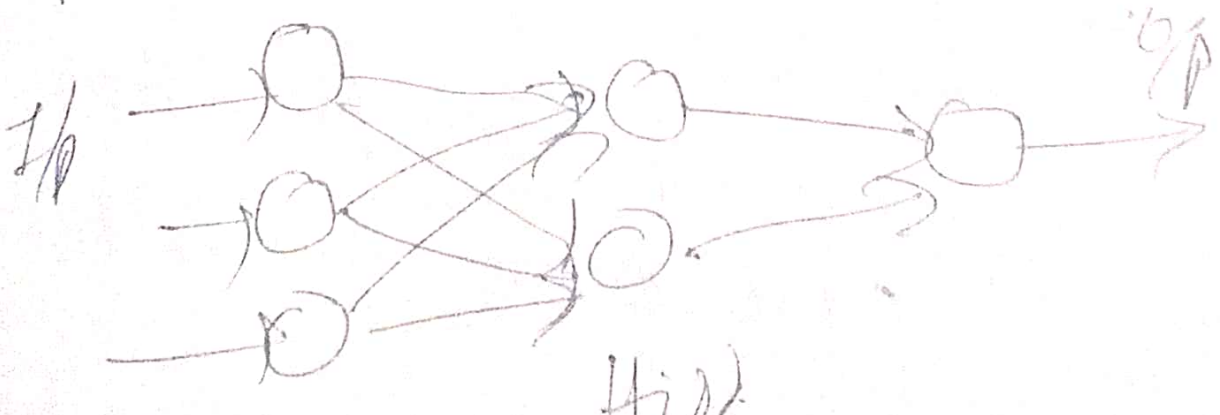
Batch size :- the number of samples processed before updating the model parameters

Type of Neural Networks

① feedforward Neural Networks

The simplest type of ANN

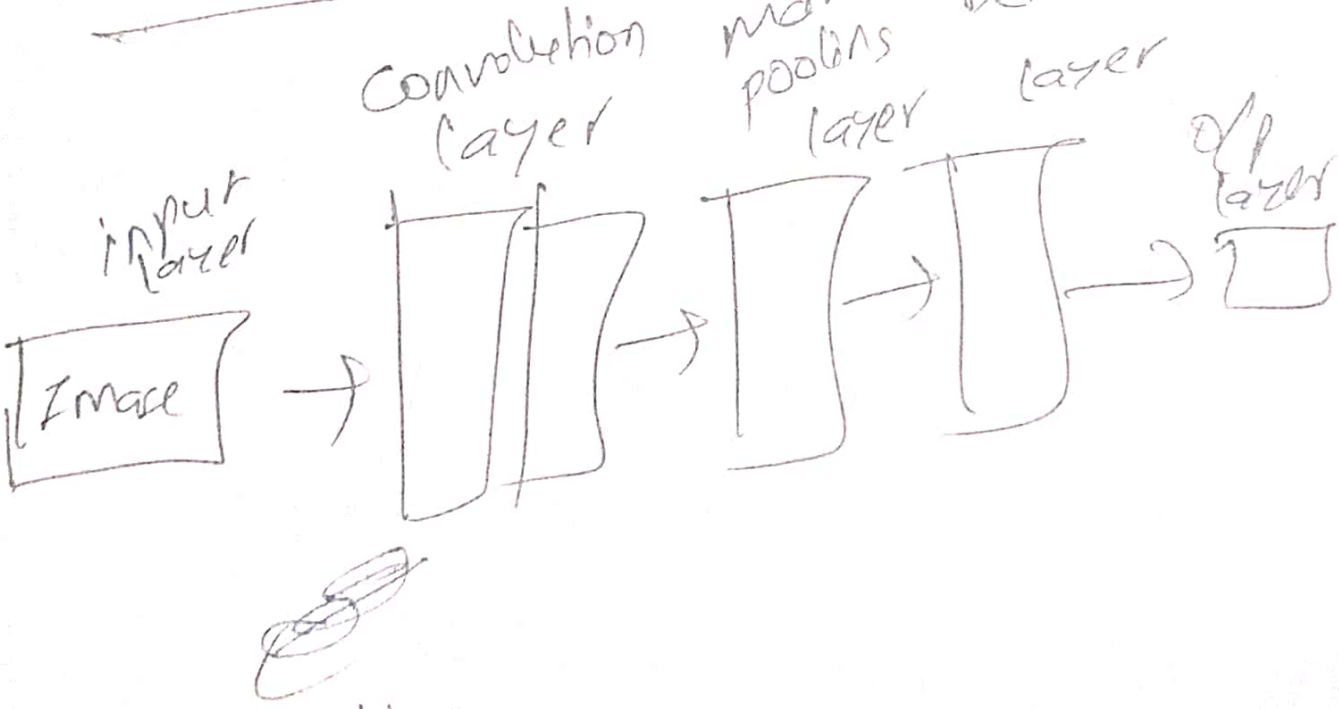
→ Data flows in one direction; from input to output.



CNN (Convolution Neural Network)

is extended version of ANN which is predominantly used to extract the features from the grid-like matrix dataset. for example visual datasets like images or videos.

CNN Architecture



convolution layer

- Extract feature from the input image using filters.

A filter (3x3) slides over the input image.

Pooling layer

Reduces the spatial dimensions of the feature map while retaining important information.

1) Reduces computational complexity
2) makes the model robust to small translation in the input.

Dense layer (fully connected layer)

Combines features extracted by previous layers to make predictions used to make final decision.

Training

① forward pass

The input image passes

through all layers to produce the output probabilities.

(2) Loss calculation

The loss is calculated by comparing the predicted probability with the true labels.

(3) Backpropagation

The gradients of the loss with respect to the weights are computed.

(4) Weight update :-

The weights are updated using an optimization algorithm.

RNN (Recurrent Neural Network)

It is a class of NN designed for processing sequential data, where the order of elements matter. unlike feedforward neural Networks, RNN have connections that form cycles, allowing them to maintain a memory of previous inputs.

* The hidden state is passed from one time step to the next, allowing the network to retain information about previous input.

Types

One to One

one input, one output.

one to many

One input, multiple outputs.

many to one

multiple inputs, one output

many to many

multiple inputs, multiple o/p-

Basic model of ANN:-

ANN are the foundation of DL and consist of inter connected nodes (neuron) organized into layer. Some basic models of ANN's are

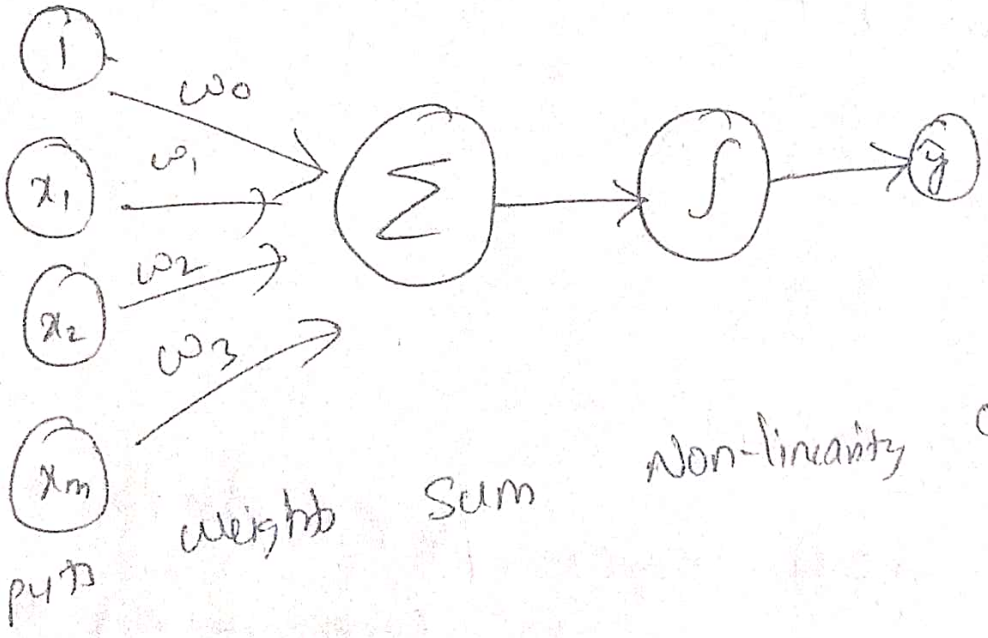
Perceptron

It is a type of artificial neuron that takes multiple binary inputs, applies weights to them, sum them up, and passes the result through an activation function to produce a binary output.

Mathematical Representation

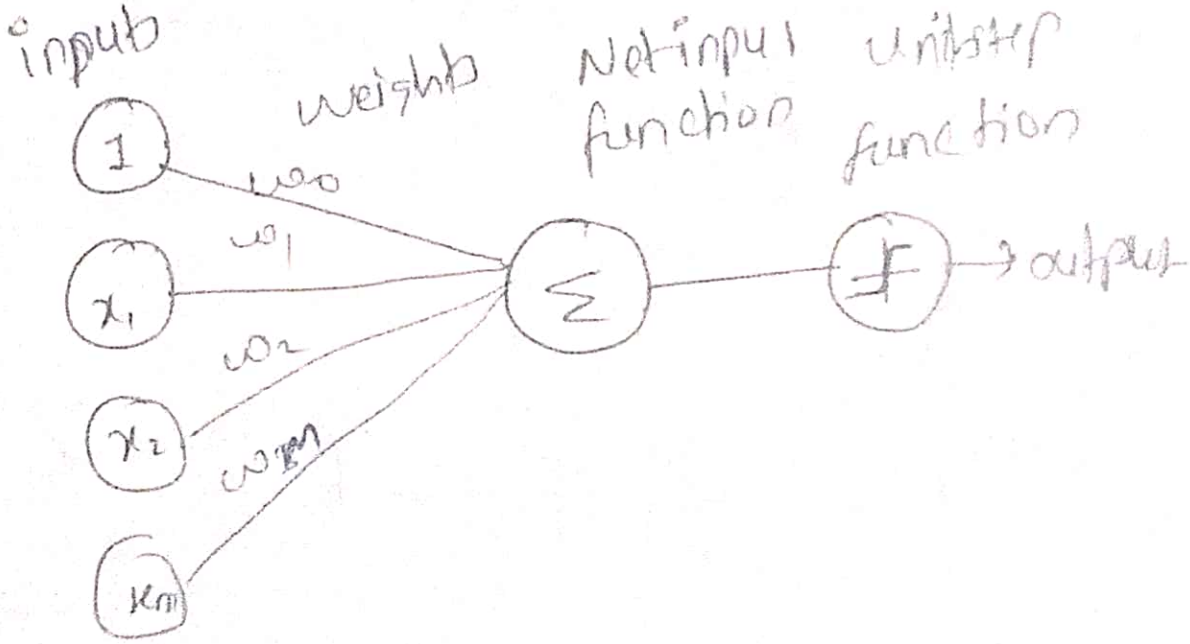
$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

- f: activation function
- w_i: weights
- x_i: input features
- b: bias.



Single-layer perceptron

A single layer perceptron consists of a single layer of output nodes, where each node is a perceptron. it is used for binary classification tasks, where the goal is to separate data into two classes.



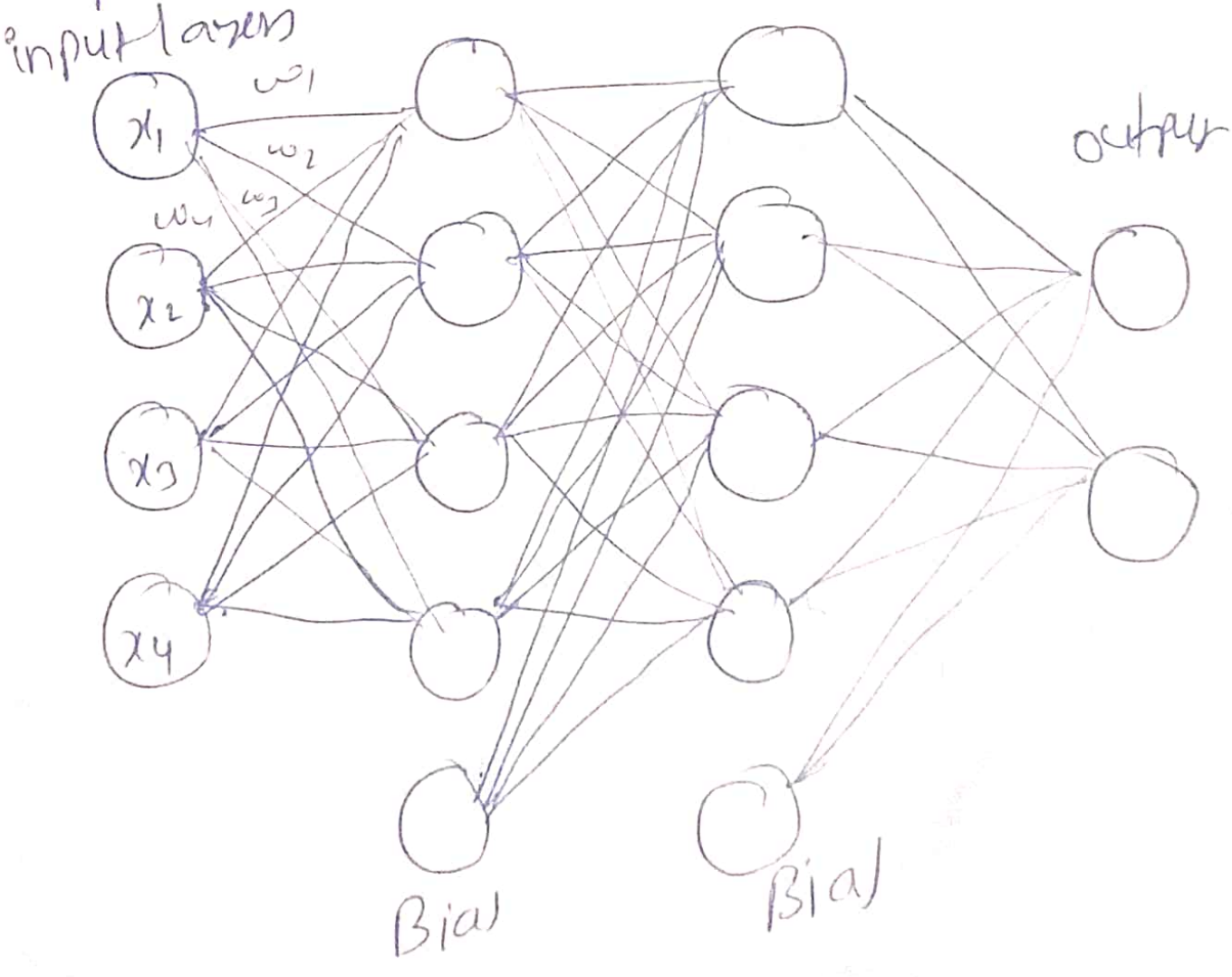
Limitations

→ single layer perceptrons only solve linearly separable problems.

→ they fail to handle more complex non linear datasets.

Multi-layer Perceptron

It is an extension of the single-layer perceptron, consisting of multiple layers of nodes (neurons). These layers include an input layer, one or more hidden layers, and an output layer. MLPs are capable of handling non-linear data and used for more complex tasks like image recognition, natural language processing, etc.



Weighted Sum =

$$\sum_{i=1}^{n} (w_i * x_i) + b$$

Advantages:

→ Capable of learning complex, non-linear relationships

→ Versatile and can be applied to a wide range of tasks.

Activation function

→ introduce non-linearity into the network, allowing it to learn complex patterns.

→ step function :- Used in basic perceptrons

→ sigmoid : smooths the output between 0 and 1

ReLU (Rectified Linear Unit) :-

Output the input directly if it is positive; otherwise, it outputs zero.

Training perceptron

→ initialization → weights are initialized randomly.

→ forward pass → inputs are passed through the network to compute the output.

→ Error calculation → The difference between the predicted and actual output is calculated.

→ Backpropagation → The error is propagated back through the network to adjust the weights.

→ weight update → weights are updated using optimization algorithm like Gradient Descent

Important technologies in ANN

① ~~Node~~ Same as the starting page. and include

Gradient Descent

→ optimization technique for minimizing loss function.

→ iteratively updates network parameters.

Loss function

→ measures prediction error.

→ common types:
Mean Squared Error.
Cross - Entropy.

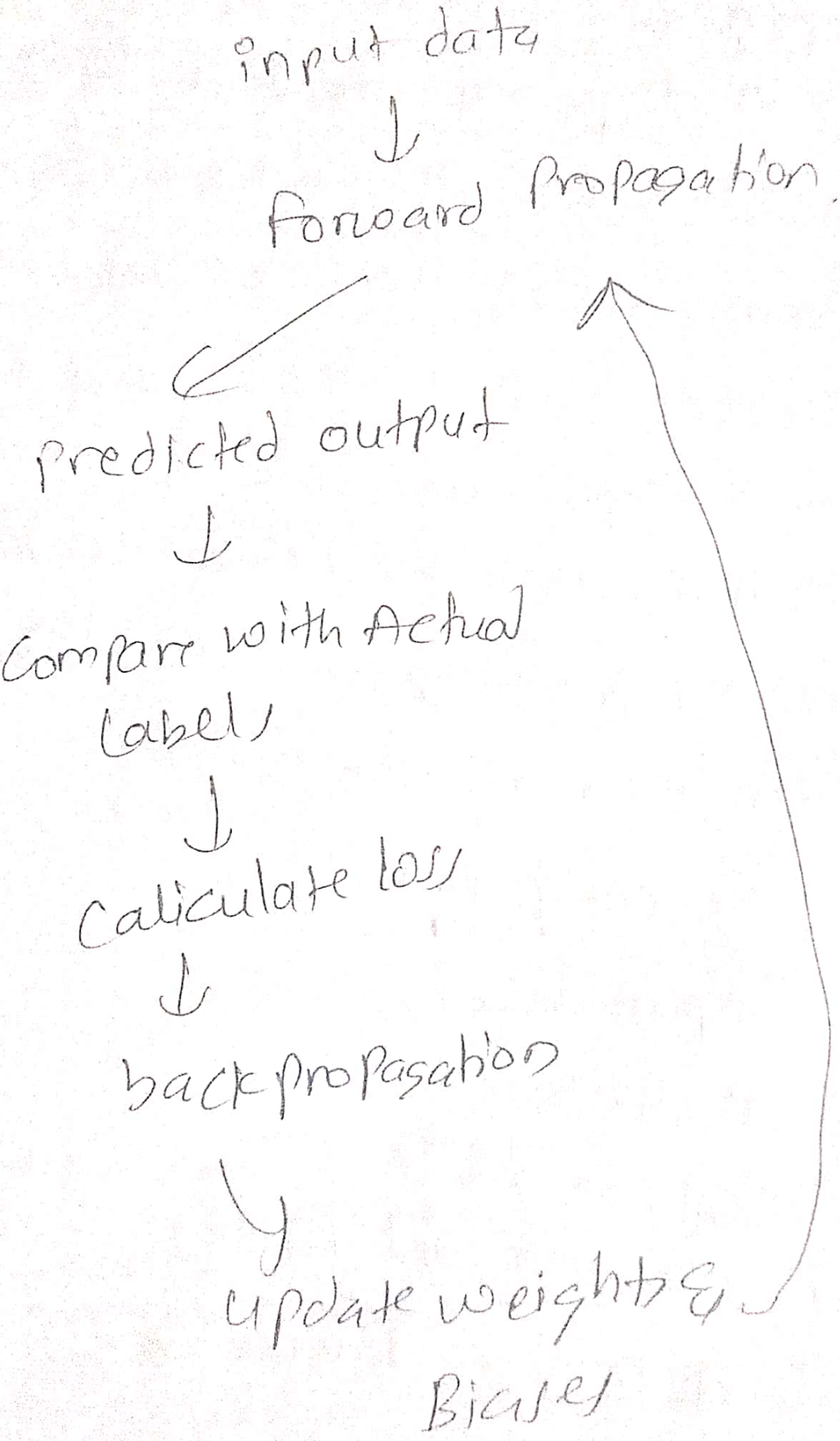
Supervised learning Networks

Supervised learning neural networks are a type of machine learning model where the network learns from labeled training data. The "supervised" part means that for each input, we know the desired output, and the network learns to map inputs to outputs by adjusting its internal parameters.

key components

- Input layer.
- Hidden layer
- Output layer.
- weights and Biases
- Activation functions.

Trainins procell



Advantages

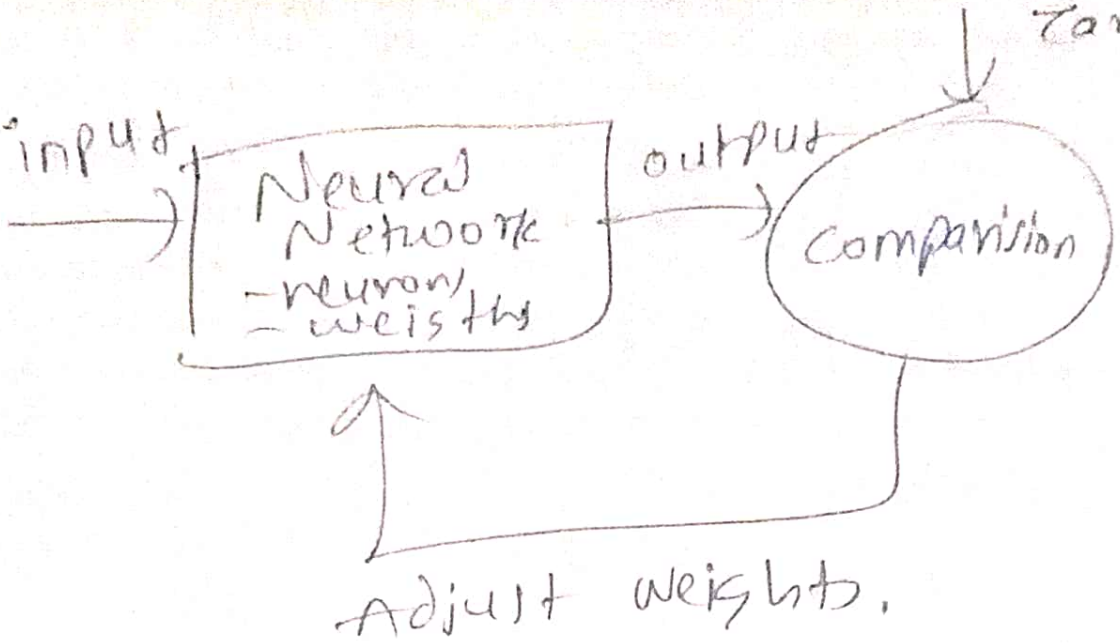
→ High Accuracy
→ Explicit feedback.
→ versatility.

Challenges

→ Labeled Data Dependency
→ overfittins.
→ computational cost.

Applications

→ medical diagnois
→ Autonomow drivins
→ Speech-to-text conversion.



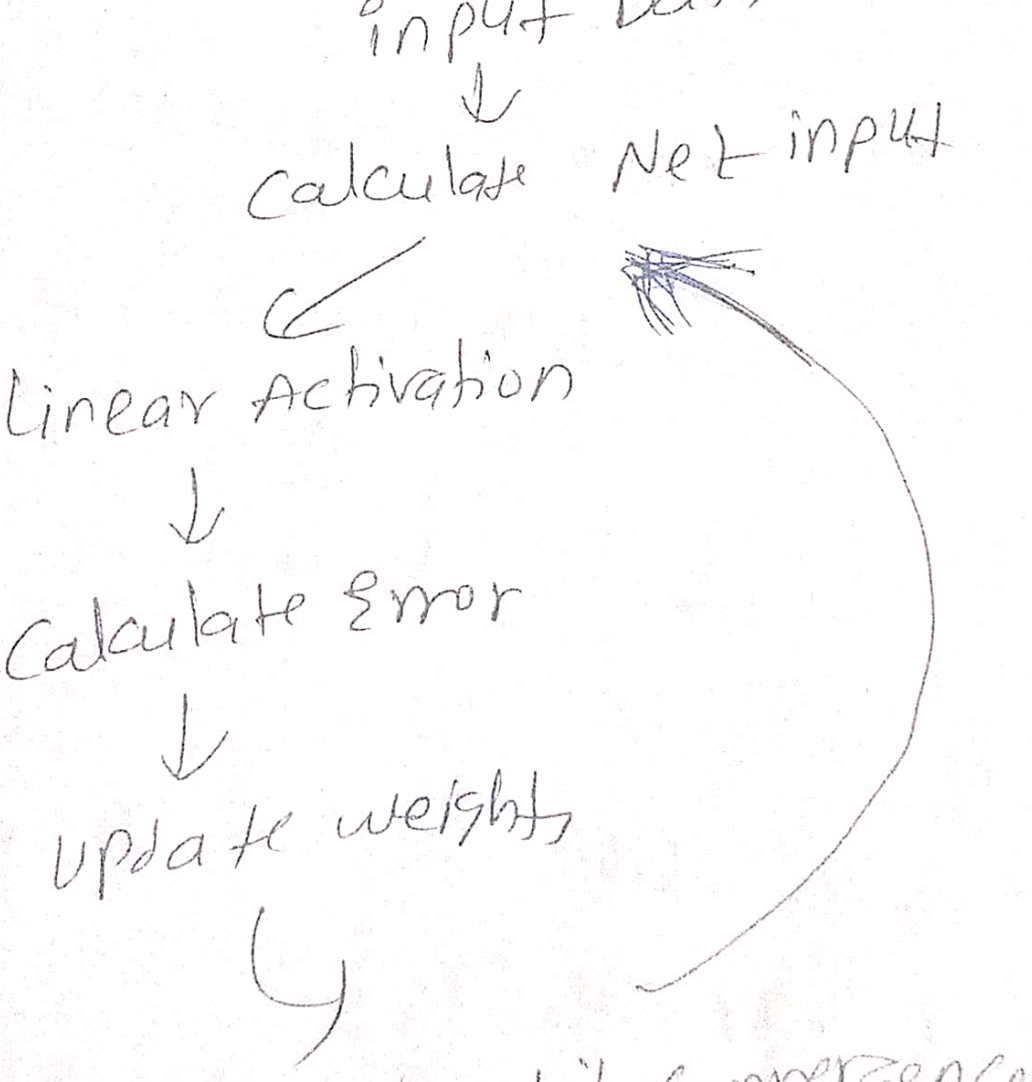
Adaptive Linear Neuron

it is a single layer neural network designed for linear regression and binary classification. unlike the perceptron, it uses a continous linear activation function during training, enabling gradient-based learning.

structure

- inputs :- Features $x_1, x_2, \ldots x_n$.
- weights :- $w_1, w_2, \ldots w_n$.
- Bias : b.
- Activation function: Linear (identity function).
- Loss function.

Learning process



learning Algorithm (Widrow - Hoff)

Delta Rule

while not converged:

for each training example (x, target):

net_input = sum(weights * input[i]) + bias

error = target - net_input

weights += learning_rate * error * input[i]

bias += learning_rate * error

Advantages

→ more stable learning than Perceptron.

Training algorithms for pattern Association

→ pattern association is a process where a neural network learns to map input patterns to corresponding output patterns. It is a fundamental concept in neural networks and is used in tasks like classification, recognition, and prediction.

Types of pattern Association

There are two types

① Auto-association:-

The network learns to map an input pattern to itself (used in tasks like noise reduction or memory recall).

2. Hetero-association

The network learn to map an input pattern to a different output pattern (used in task like classification).

3. Trainins Algorithms for Pattern Association

The molt common algorithms are.

① Hebbian Learnins Rule.
② Perceptron Learnins Rule.
③ Delta Rule (Widrow-Hoff Rule)
④ Back propagation Algorithms

① Hebbian Learnins Rule

→ Cells that fire touse ther, wire together.

→ It strengthens the connection between neurons if they are activated simultaneously.

Algorithm

1. Initialize weights to zero.
2. for each input-output pair (x, y):
   → update weights:-

$$\Delta w_{ij} = \eta \cdot x_i \cdot y_j$$

where:-

η = learning rate.

x_i = input

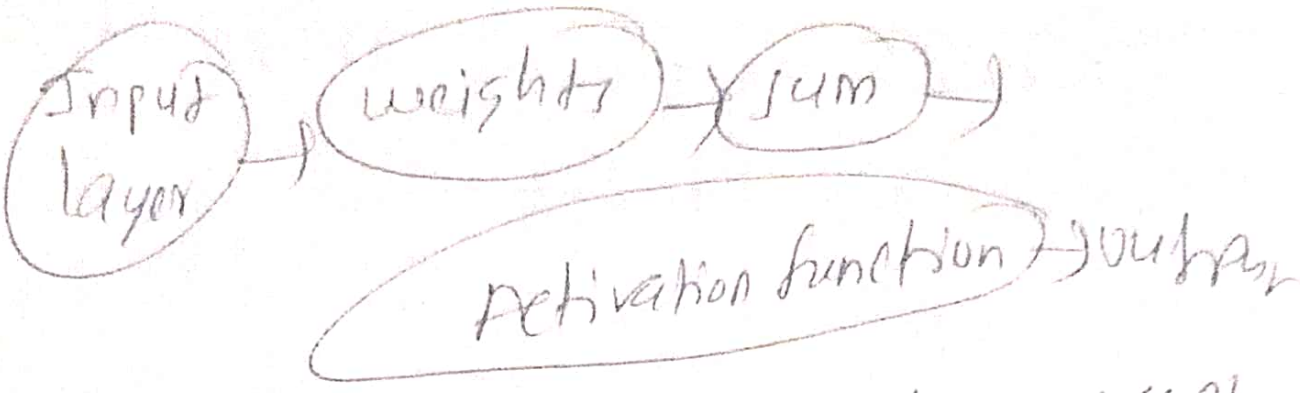
y_j = output

3. Repeat until convergence.

(2) perceptron Learning Rule

→ used for binary classification tasks.

→ Adjusts weights based on the error between the predicted output and the target output.



Same we write about perceptron.

(3) Delta Rule

→ A generalization of the perceptron learning Rule.

→ minimizes the error between the actual output and the target output

input → wights → sum → output.

also known as (Widrow - Hoff Rule)

Backpropagation

Input layer → Hidden layer → output layer

forward pass : $x \rightarrow w_1 \rightarrow H \rightarrow w_2 \rightarrow Y$

backward pass : Error $\rightarrow w_2 \rightarrow H \rightarrow w_1 \rightarrow x$

Alg	type	key formula	use
Hebbian learning	unsupervised	$\Delta w_{ij} = \eta \cdot x_i \cdot x_j$	Pattern recall, memory
Perceptron learning	supervised	$\Delta w_i = \eta \cdot (y - y_{pred}) \cdot x_i$	Binary classification
Delta Rule	supervised	$\Delta w_i = \eta \cdot (y - y_{pred}) \cdot x_i$	Linear regression, prediction
Backpropagation	supervised	$\Delta w_{ij} = \eta \cdot \delta_j \cdot x_i$	multi-layer networks, deep learning

Associative memory Network

It refers to the ability to recall a complete pattern or memory when presented with a partial or noisy version of it.

Ex:- if you see a blurry image of a cat, your brain can still recognize it as a cat. So same AMN also mimic this.

How it works

It store patterns (image, words, data) in a way that allows them to retrieve the full pattern even when given an incomplete or distorted version.

Storage phase

During training, the network learns to store patterns by adjusting the weights of the connection between neurons. These weights represent the associations between different parts of patterns.

Retrieval phase

When a partial or noisy input is provided, the network uses the stored weights to reconstruct the complete pattern. This done through iterative updates until the network converges to the closet stored pattern.

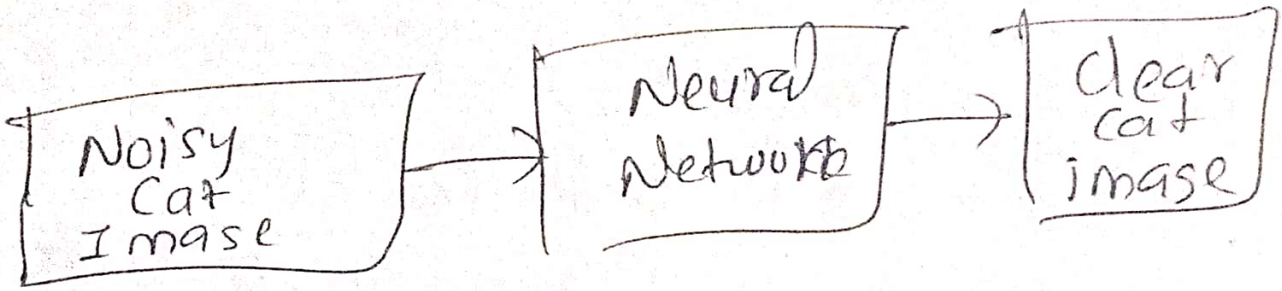
Types of ANN

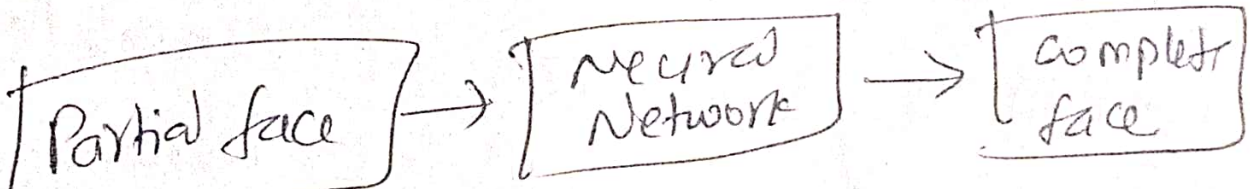
1. Autoassociative memory
2. Heteroassociative memory.

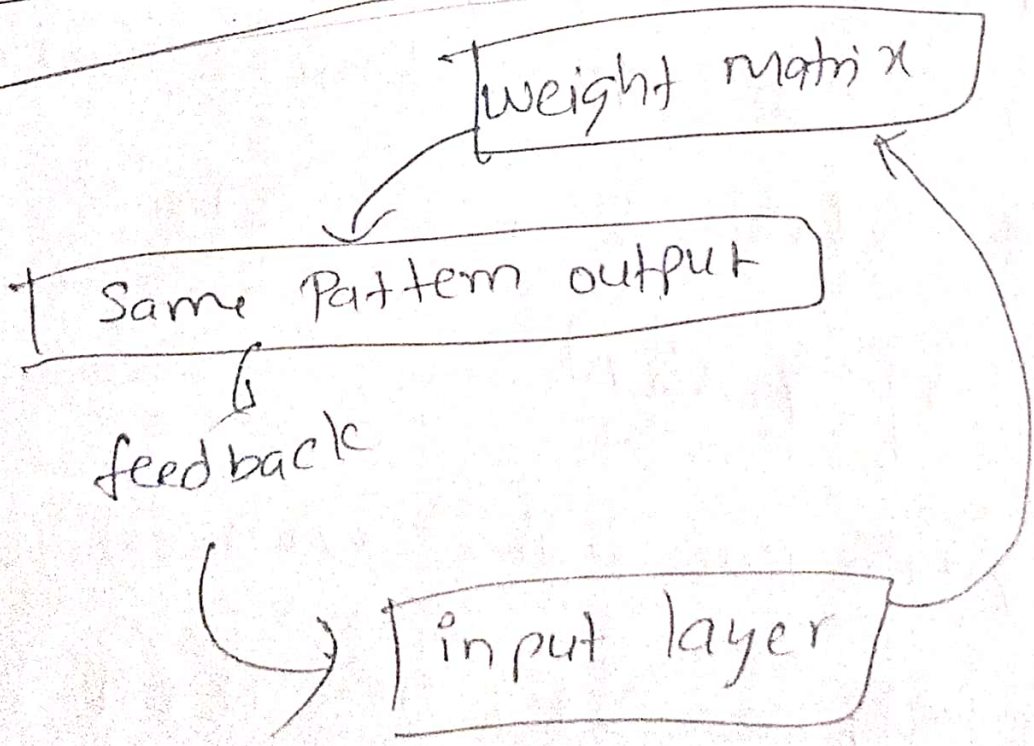
(1) Autoassociative memory

→ Stores and retrieves the same type of pattern.

→ Used for tasks like image denoising or error correction.



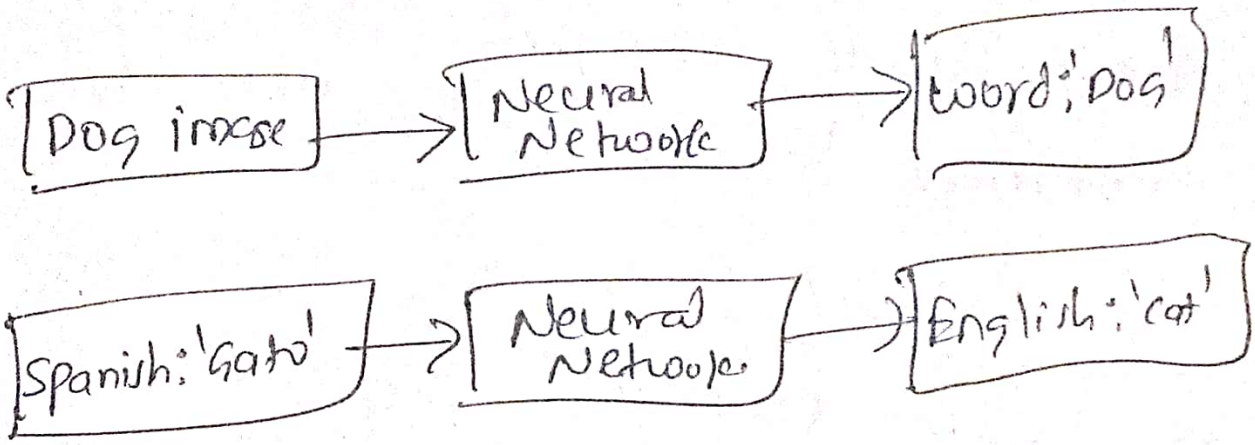


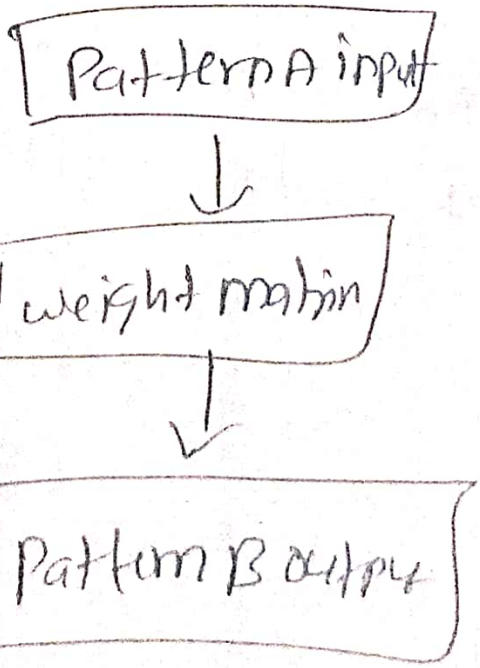


② Heteroassociative memory

→ stores associations between different types of patterns.

→ used for tasks like mapping inputs to outputs (eg word-to-word retrieval)





Applications

1. Image and speech recognition.
2. Data Retrieval.
3. Error correction.

Advantages

1. Robust to noise and errors.
2. Can handle incomplete data.
3. Mimics human memory process.

Limitations

1. Limited storage capacity.
2. computationally expensive.
3. May retrieve incorrect patterns if input is too noisy.

Back-Propagation Network

Back-propagation is a fundamental alg used to train neural networks. it is a supervised learning technique that adjusts the weights of the network to minimize the error between the predicted output and the actual ~~outp~~ target.

How it works

it has two main phases

forward pass

→ input data is fed into the network.

→ The network computes the output using the current weights and activation function.

→ The error is calculated using a loss function.

(2) Backward pass

→ the error is propagated backward through the network.

→ the gradient of the error with respect to each weight is computed using the chain rule of calculus.

→ The weights are updated to minimize the error using gradient descent

Steps

(1) initialize weights.

(2) forward pass
   → compute the output of each layer
   → calculate loss (error) at output layer

(3) Backward pass.
→ compute the gradient of the loss with respect to each weight.

→ Update the weights using the formula:

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

w_{new} → updated weight

w_{old} → current weight

η → learning rate

$\frac{\partial L}{\partial w}$ → Gradient of the loss with respect to weight

(4) Repeat → until the error is minimized.

Advantages

→ can learn complex, non-linear relationships in data.

→ works well with large datasets.

→ highly versatile and applicable to various tasks.

Limitations

→ computationally expensive.
→ Requires careful tuning of hyperparameters (learning rate)

BAM (Bidirectional Associative Memory) and Hopfield Networks

both used in neural networks. They are designed to store and retrieve patterns, making them useful for tasks like pattern recognition, data retrieval, and error correction.

① Hopfield Network

It is type of Recurrent Neural Network that serves as a content-addressable memory system. It can store and retrieve patterns

based on partial or noisy inputs.

Key features

→ Recurrent Architecture
  ↳ Neurons are connected in a feedback loop, meaning the output each neuron is fed back as input to other neuron.

→ Energy function
  ↳ The network has an energy function that decreases over time, ensuring convergence to a stable state.

→ Autoassociative memory.
  ↳ it can retrive a complete pattern when given a partial or noisy version of it.

How it works

① storage phase.

② ~~Retrieval~~ Retrieval phase.

① storage phase

Patterns are stored in the network using the weights between neurons using Hebbian learning.

$$w_{ij} = \frac{1}{N} \sum_{p=1}^{P} x_i^p x_j^p$$

- w_{ij}: weight between i and neuron j
- N: Number of neurons
- P: Number of patterns
- x_i^p: value of neuron i in pattern p.

② Retrieval phase

given a partial or noisy input, the network iteratively updates the neurons states until it converges to the ~~dot~~ closest stored pattern.

Applications

(1) Pattern completion.
(2) optimization problems.

Ex:

let store the pattern $[1, -1, 1]$ in a Hopfield Network.

(1) pattern Matrix is

$$x = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$$

(2) Compute the weight matrix.

$$W = \frac{1}{N} x \cdot x^T$$

$N = 3$ (number of neurons)

x^T is transpose of x

$$x \cdot x^T = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & -1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{bmatrix}$$

$$W = \frac{1}{3}\begin{bmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{bmatrix}$$

(3) Retrieve the pattern.

Suppose we have a noisy input $[1, -1, -1]$

$$x_{input} = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}$$

multiply the weight matrix with input

$$W \cdot x_{input} = \frac{1}{3}\begin{bmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}$$

$$= \frac{1}{3}\begin{bmatrix} 1\cdot 1 + (-1)(-1) + 1\cdot(-1) \\ (-1)\cdot 1 + 1\cdot(-1) + (-1)\cdot(-1) \\ 1\cdot 1 + (-1)\cdot(-1) + 1\cdot(-1) \end{bmatrix}$$

$$= \frac{1}{3}\begin{bmatrix} 1 + 1 - 1 \\ -1 - 1 + 1 \\ 1 + 1 - 1 \end{bmatrix}$$

$$= \frac{1}{3}\begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$$

Apply the sign function and the final output is

$$x_{output} = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$$

It retrives the original pattern.

$[1, -1, 1]$

(2) BAM (Bidirectional Associative memory)

BAM is extension of Hopfield Network that allows for hetero associative memory. it can store and retrieve association between two different set of patterns.

key-features

→ Bidirectional Architecture
  ↳ has two layers that are fully connected both directions.

→ Hetroassociation memory
  ↳ It can map patterns from one set to another.

→ Energy function
  ↳ Like the Hopfield Network, BAM has an energy function that ensures convergence

How it works

① Storage phase

Patterns are stored by adjusting the weights between the two layers using Hebbian learning.

$$w_{ij} = \sum_{p=1}^{P} x_i^p y_j^p$$

w_{ij} → weight between neuron i in layer A and neuron j in layer B.

$x_i^p y_j^p$ → value

(2) Retrieval phase

Given an input in one layer, the network retrieves the associated pattern in the other layer by iteratively updating the neuron states.

Applications

→ Associative recall.
→ pattern recognition and data retrieval.

Limitations

→ Limited storage capacity

Ex:-

imagine you store the association between [1, -1] in layer A and [1, 1] in layer B.

(1) Step-1: create the pattern Matrices.

Layer A pattern $x = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$

Layer B pattern $y = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

(2) compute weights

$$W = x \cdot y^T$$

compute y^T:

$$y^T = \begin{bmatrix} 1 & 1 \end{bmatrix}$$

$$W = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 \cdot 1 & 1 \cdot 1 \\ -1 \cdot 1 & -1 \cdot 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}$$

(3) Retrieve the pattern

Suppose the input layer A is $[1, -1]$

$$x_{input} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

multiply the weight matrix W^T with input vector

$$W^T = \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}$$

$$W^T \cdot x_{input} = \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

$$= \begin{bmatrix} 1.1 + (-1).(-1) \\ 1.1 + (-1).(-1) \end{bmatrix}$$

$$= \begin{bmatrix} 1+1 \\ 1+1 \end{bmatrix}$$

$$= \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

after Applying sign function

output

$$y_{output} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

The network retrieved the associated Pattern $[1, 1]$ in layer B.

Sign function working

+1 if input is positive

-1 if input is negitive

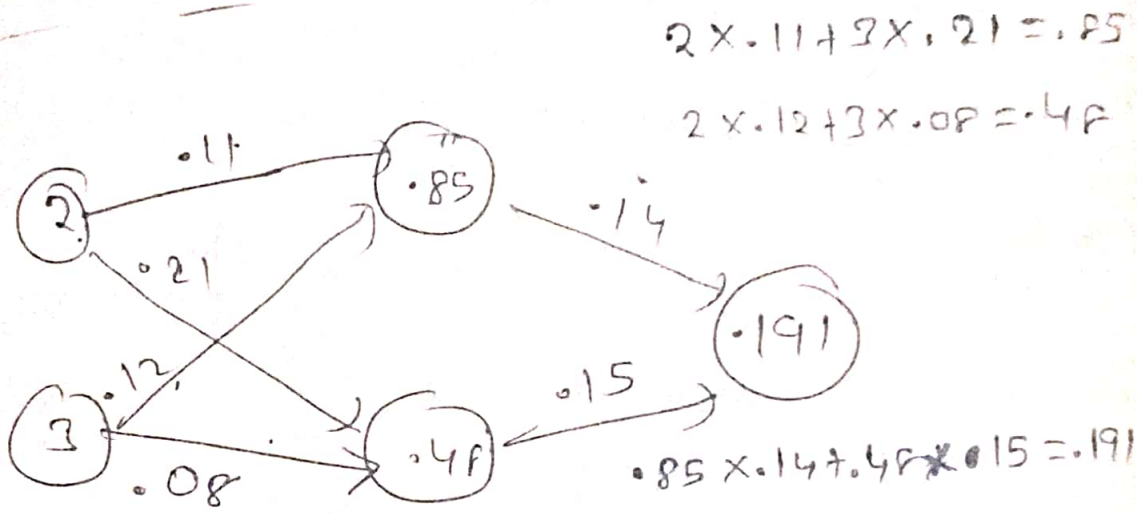
0 if input is 0

Dataset

Our single sample is as following

input = [2, 3] and output = 1

Forward Pass



$$[2 \quad 3] \cdot \begin{bmatrix} 0.11 & 0.12 \\ 0.21 & 0.08 \end{bmatrix} = [0.85 \quad 0.46] \cdot \begin{bmatrix} 0.14 \\ 0.15 \end{bmatrix}$$

$$= [0.191]$$

Calculating Error

$$\text{Error} = \frac{1}{2}(\text{Prediction} - \text{acutal})^2$$

$$= \frac{1}{2}(0.191 - 1.0)^2 = 0.327$$

Backpropagation

* Backpropagation short for "backward propagation of errors", is a mechanism used to update the weights using gradient descent.

* It calculates the gradient of the error function with respect to the neural network's weights.

* The calculation proceeds backwards through the network

$$w_x = w_x - a\left(\frac{\partial \text{Error}}{\partial w_x}\right)$$

(Derivate Error with respect to weight)

w_x ← new weight; w_x ← old weight; a ← learning rate

* $w_6 = w_6 - a(h_2 \cdot \Delta)$
* $w_5 = w_5 - a(h_1 \cdot \Delta)$
* $w_4 = w_4 - a(i_2 \cdot \Delta w_6)$

* $w_3 = w_3 - a(i_1 \cdot \Delta w_6)$

* $w_2 = w_2 - a(i_1 \cdot \Delta w_5)$

* $w_1 = w_1 - a(i_1 \cdot \Delta w_5)$

$$\begin{bmatrix} w_5 \\ w_6 \end{bmatrix} = \begin{bmatrix} w_5 \\ w_6 \end{bmatrix} - a\Delta \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = \begin{bmatrix} w_5 \\ w_6 \end{bmatrix} - \begin{bmatrix} ah_1\Delta \\ ah_2\Delta \end{bmatrix}$$

$$\begin{bmatrix} w_1 & w_3 \\ w_2 & w_4 \end{bmatrix} = \begin{bmatrix} w_1 & w_3 \\ w_2 & w_4 \end{bmatrix} - a\Delta \begin{bmatrix} i_1 \\ i_2 \end{bmatrix}.$$

$$\begin{bmatrix} w_5 & w_6 \end{bmatrix} = \begin{bmatrix} w_1 & w_3 \\ w_2 & w_4 \end{bmatrix} - \begin{bmatrix} a i_1 \Delta w_7 & a i_1 \Delta w_8 \\ a i_2 \Delta w_5 & a i_2 \Delta w_6 \end{bmatrix}$$

Backward Pass

$\Delta = 0.191 - 1 = -0.809$

$a = 0.05$.

$$\begin{bmatrix} w_7 \\ w_8 \end{bmatrix} = \begin{bmatrix} 0.14 \\ 0.15 \end{bmatrix} - 0.05(-0.809)$$

$$\begin{bmatrix} 0.85 \\ 0.91 \end{bmatrix}$$

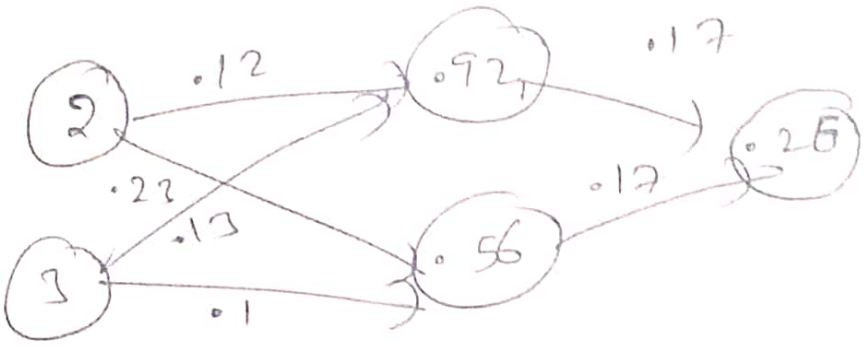
$$= \begin{bmatrix} 0.14 \\ 0.15 \end{bmatrix} - \begin{bmatrix} -0.074 \\ -0.019 \end{bmatrix} = \begin{bmatrix} 0.17 \\ 0.17 \end{bmatrix}$$

$$\begin{bmatrix} w_1 & w_3 \\ w_2 & w_4 \end{bmatrix} = \begin{bmatrix} .11 & .12 \\ .21 & .01 \end{bmatrix} - 0.05(-0.509)$$

$$\begin{bmatrix} 2 \\ 2 \end{bmatrix} \cdot \begin{bmatrix} 0.14 & 0.15 \end{bmatrix}$$

$$= \begin{bmatrix} .11 & .12 \\ .21 & .01 \end{bmatrix} - \begin{bmatrix} -0.011 & -0.012 \\ -0.017 & -0.014 \end{bmatrix}$$

$$= \begin{bmatrix} .12 & .13 \\ .23 & .10 \end{bmatrix}$$



$$\begin{bmatrix} 2 & 3 \end{bmatrix} \cdot \begin{bmatrix} 0.12 & 0.13 \\ 0.23 & 0.10 \end{bmatrix} = \begin{bmatrix} 0.92 & 0.56 \end{bmatrix} \cdot \begin{bmatrix} 0.17 \\ 0.17 \end{bmatrix}$$

$$= [0.26]$$

* we notice that the Prediction 0.26 is a little bit closer to actual output that the previously Predicted one 0.191