23EC704: ARTIFICIAL NEURAL NETWORKS

TOPIC :INTRODUCTION TO ARTIFICIAL NEURAL NETWORKS

.

MR: A DILEEP

Assistant Professor,

ELECTRONICS AND COMMUNICATION ENGINEERING

Narsimha Reddy Engineering College (Autonomous)

Secunderabad, Telangana, India- 500100

UNIT 1
INTRODUCTION

CONTENTS

- Neural network
- Neural Networks and Human Brain
- Models of a Neuron
- Neural Networks viewed as Directed Graphs
- Network Architectures
- Knowledge Representation
- Artificial Intelligence in Neural Networks

CONTENTS

- Learning Process
- Credit Assignment Problem
- Statistical Nature of the Learning Process

WHAT IS A NEURAL NETWORK?

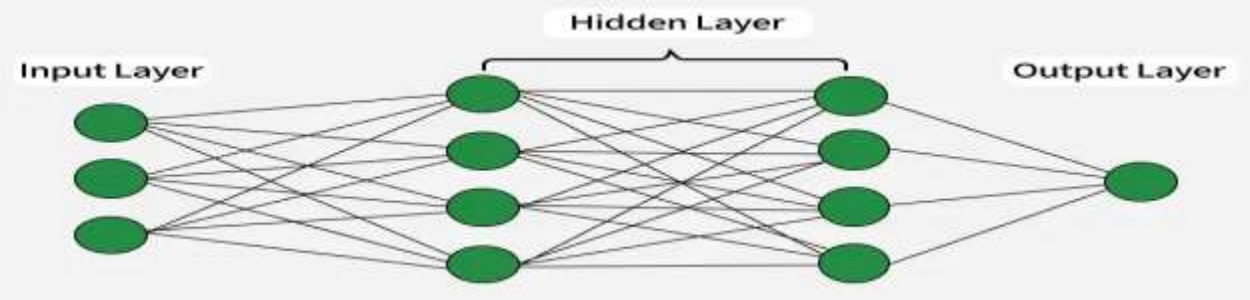
- An artificial neural network is a model inspired by the human brain, built from interconnected nodes ("neurons") that process information and learn patterns from data.
- Inspired by the structure of the human brain
- Composed of artificial neurons (nodes) and weighted connections
- Used in AI for pattern recognition, learning, and decision-making

KEY CONCEPTS OF NEURAL NETWORKS

- Artificial Neurons (Nodes)
- Layers
- Weights & Biases
- Activation Functions
- Learning Rule (Training)

KEY CONCEPTS OF NEURAL NETWORKS

- **Artificial Neurons (Nodes):** Simplified versions of biological neurons. Each receives inputs, applies a mathematical function, and produces an output.



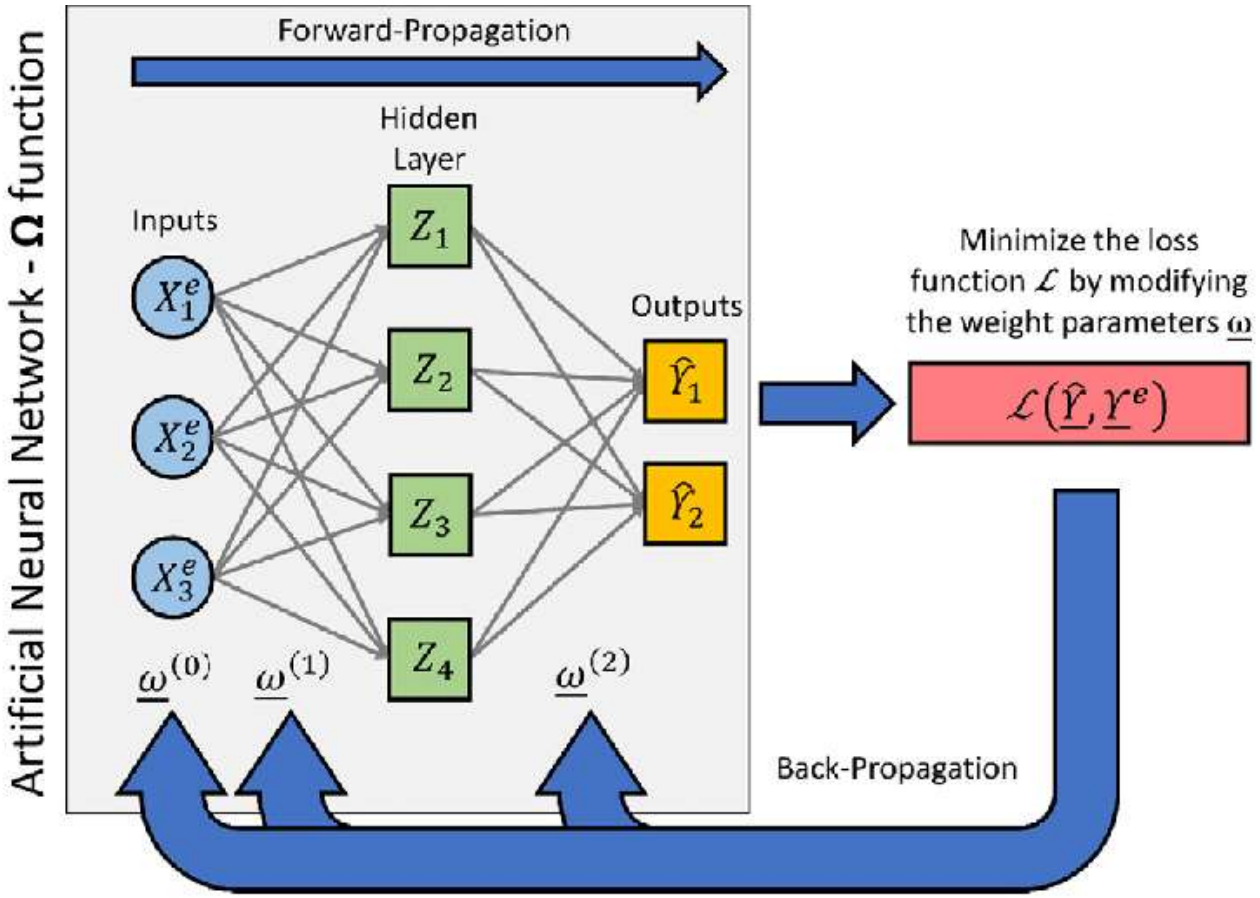
Layers:

- **Input Layer:** Receives raw data (e.g., pixels in an image).
- **Hidden Layers:** Transform inputs through weighted connections and activation functions. ie transforms the inputs into something that the output layer can use.
- **Output Layer:** Produces the final prediction or classification.

KEY CONCEPTS OF NEURAL NETWORKS

- **Weights & Biases:** Parameters that determine the strength and influence of connections between neurons. They are adjusted during training to improve accuracy.
- **Activation Functions:** Decide whether a neuron should "fire" (e.g., sigmoid, ReLU). They introduce non-linearity, allowing networks to model complex relationships.
- **Learning Rule (Training):** Uses algorithms like back propagation to update weights and biases based on errors, improving performance over time.

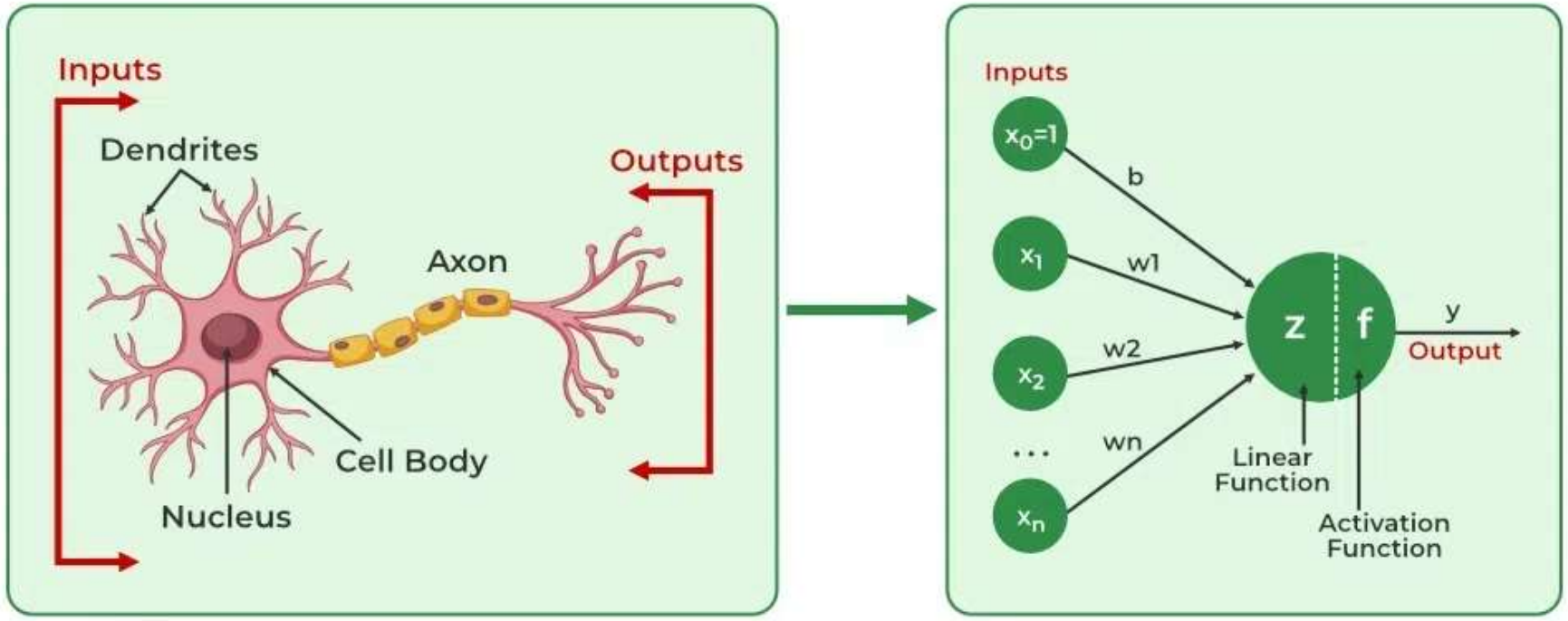
HOW NEURAL NETWORKS WORK



HOW NEURAL NETWORKS WORK

- **Data Input:** Information enters through the input layer.
- **Forward Propagation:** Data flows through hidden layers, transformed by weights and activation functions.
- **Output Prediction:** The network produces a result (e.g., "cat" vs. "dog").
- **Error Calculation:** The difference between predicted and actual output is measured.
- **Backpropagation:** The network adjusts weights to minimize error, repeating until performance stabilizes.

HUMAN BRAIN VS NEURAL NETWORK



HUMAN BRAIN VS NEURAL NETWORK

Aspect	Biological Neurons (Brain)	Artificial Neural Networks (AI)
Basic Unit	Neuron (cell)	Node (mathematical function)
Communication	Electrical impulses	Weighted numerical values
Learning Mechanism	Synaptic plasticity	Backpropagation & optimization
Speed	Milliseconds	Microseconds (digital)
Scale	~86 billion neurons	Thousands to millions of nodes

MODELS OF ARTIFICIAL NEURAL NETWORK(NEURON)

- MCCULLOCH-PITTS MODEL
- ROSENBLATT'S PERCEPTRON MODEL
- ADALINE NETWORK MODEL

MODELS OF ARTIFICIAL NEURAL NETWORK(NEURON)

1. McCulloch-Pitts Model

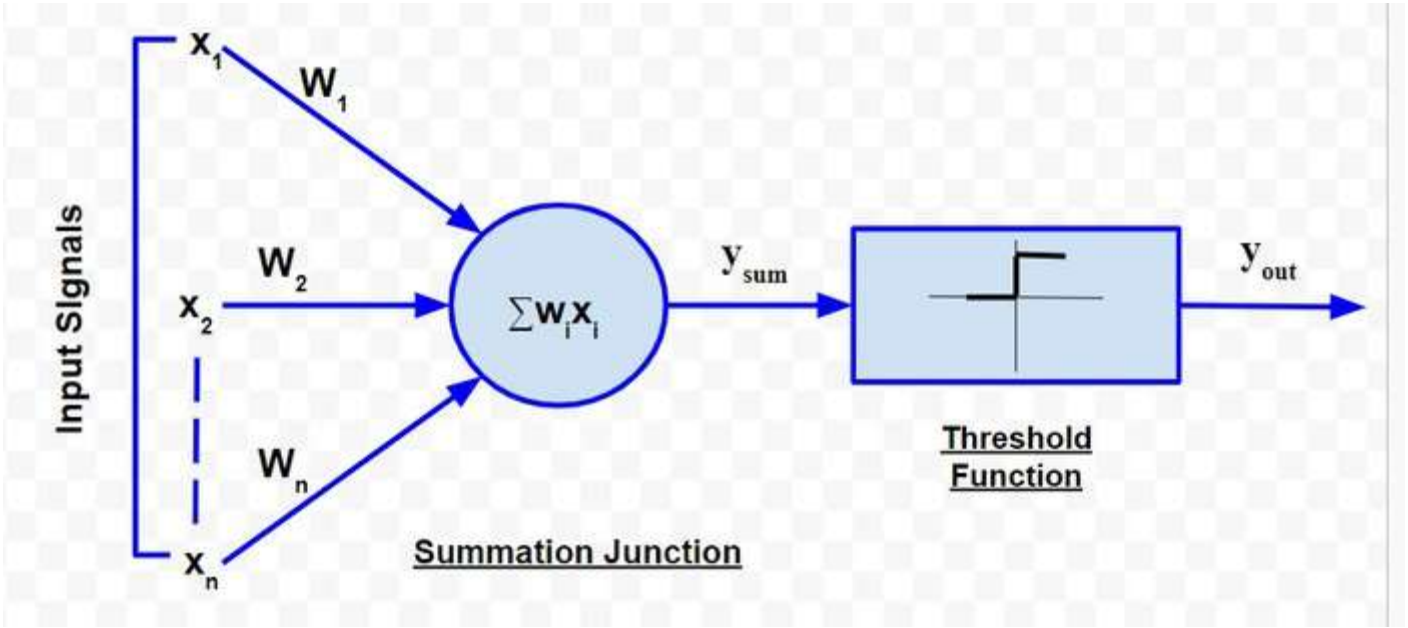
Types of inputs — Excitatory and Inhibitory.

- Excitatory inputs have weights of positive magnitude

Inhibitory weights have weights of negative magnitude.

- Inputs 0 or 1.
- Threshold function as an activation function.
- Output signal Yout is 1 if the input y sum is greater than or equal to a given threshold value, else 0.

MCCULLOCH-PITTS MODEL



MCCULLOCH-PITTS MODEL

- McCulloch-Pitts neurons can be used to design logical operations
- consider an example:ARUN carries an umbrella if it is sunny or if it is raining. There are four given situations
- when ARUN will carry the umbrella. The situations are as follows:
- First scenario: It is not raining, nor it is sunny
- Second scenario: It is not raining, but it is sunny
- Third scenario: It is raining, and it is not sunny
- Fourth scenario: It is raining as well as it is sunny

MCCULLOCH-PITTS MODEL

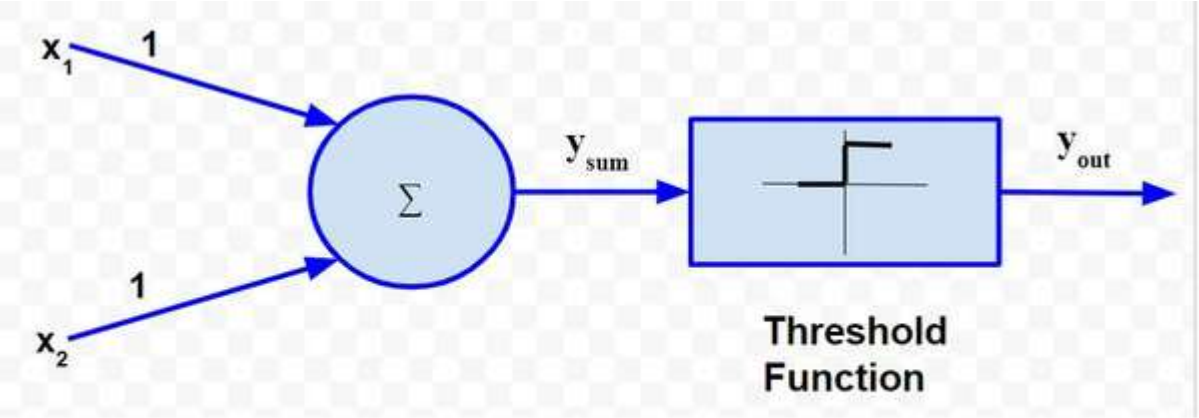
- To analyse the situations using the McCulloch-Pitts neural model, I can consider the input signals as follows:
- X_1: Is it raining?
- X_2 : Is it sunny?

So, the value of both scenarios can be either 0 or 1.

We can use the value of both weights X_1 and X_2 as 1 and a threshold function as 1.

So, the neural network model will look like:

MCCULLOCH-PITTS MODEL



MCCULLOCH-PITTS MODEL

Truth Table for this case will be:

Situation	x_1	x_2	y_{sum}	y_{out}
1	0	0	0	0
2	0	1	1	1
3	1	0	1	1
4	1	1	2	1

MCCULLOCH-PITTS MODEL

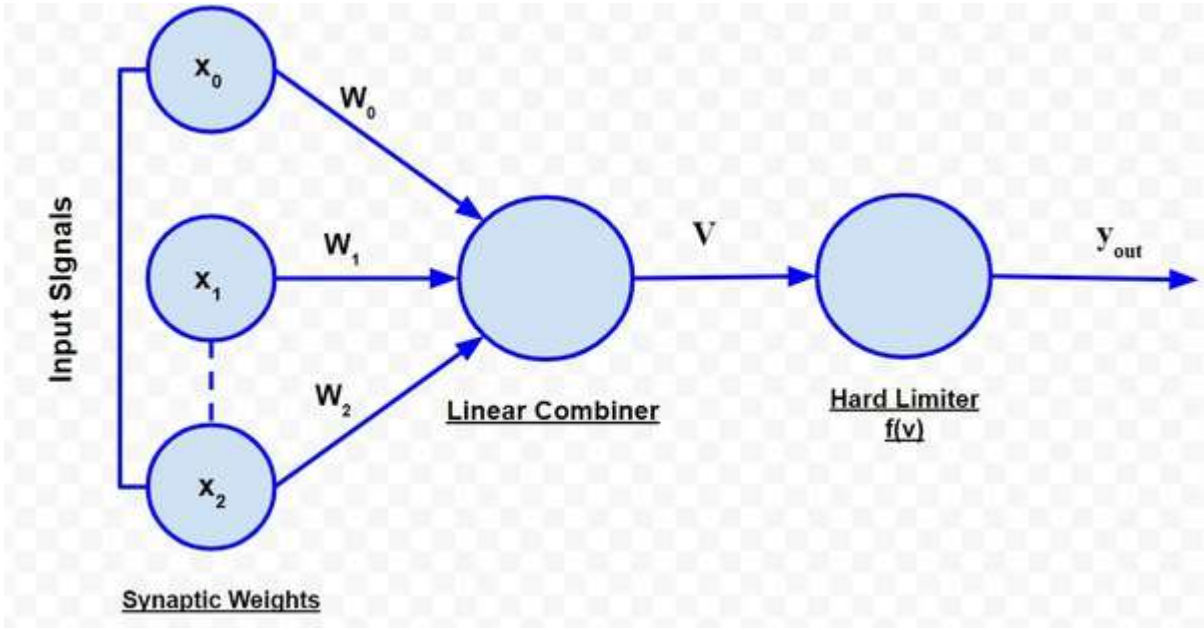
From the truth table, I can conclude that in the situations where the value of yout is 1, arun needs to carry an umbrella. Hence, he will need to carry an umbrella in scenarios 2, 3 and 4.

ROSENBLATT'S PERCEPTRON MODEL

- The perceptron receives a set of input x_1, x_2,....., x_n.
- The linear combiner computes the linear combination of the inputs applied to the synapses with synaptic weights being w_1, w_2,.....,w_n.
- The hard limiter checks whether the resulting sum is positive or negative.
- If the input of the hard limiter node is positive, the output is +1, and if the input is negative, the output is -1.
- Mathematically the hard limiter input is:

$$v = \sum_{i=1}^{n} w_i x_i$$

ROSENBLATT'S PERCEPTRON MODEL



ROSENBLATT'S PERCEPTRON MODEL

- However, perceptron includes an adjustable value or bias as an additional weight w_0.
- This additional weight is attached to a dummy input x_0, which is assigned a value of 1.
- This consideration modifies the above equation to:

$$v = \sum_{i=0}^{n} w_i x_i$$

- The output is decided by the expression:

$$y_{out} = f(v) = \begin{cases} +1, v > 0 \\ -1, v < 0 \end{cases}$$

ROSENBLATT'S PERCEPTRON MODEL

- The objective of the perceptron is to classify a set of inputs into two classes c_1 and c_2.
- assign the inputs to
- c_1 if y_{out} is +1
- c_2 if y_{out} is -1.

.

ROSENBLATT'S PERCEPTRON MODEL

- For an n-dimensional signal space i.e. a space for 'n' input signals
- The simplest form of perceptron will have two decision regions, resembling two classes, separated by a hyperplane defined by:

$$\sum_{i=0}^{n} w_i x_i = 0$$

- Therefore, the two input signals denoted by the variables x_1, x_2 and the decision boundary is a straight line of the form:

$$w0.x0+w1.x1+w2.x2=0$$
$$w0+w1.x1+w2.x2=0[\because x0=1]$$

ROSENBLATT'S PERCEPTRON MODEL

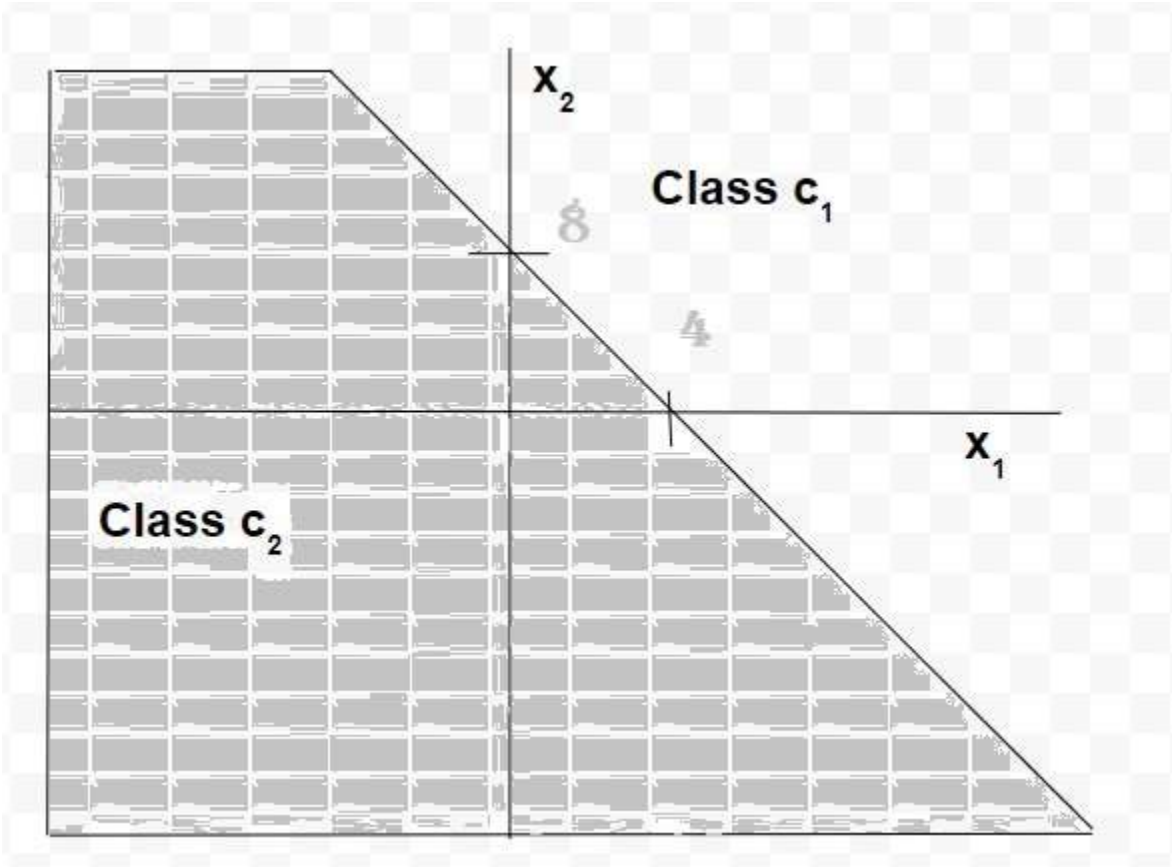
- So, for a perceptron having the values of synaptic weights w_0,w_1 and w_2 as -2, 1/2 and 1/4, respectively. The linear decision boundary will be of the form:

$$-2 + \tfrac{1}{2}x_1 + \tfrac{1}{4}x_2 = 0$$

$$-2 + \tfrac{1}{2}x_1 + \tfrac{1}{4}x_2 = 2x_1 + x_2 = 8$$

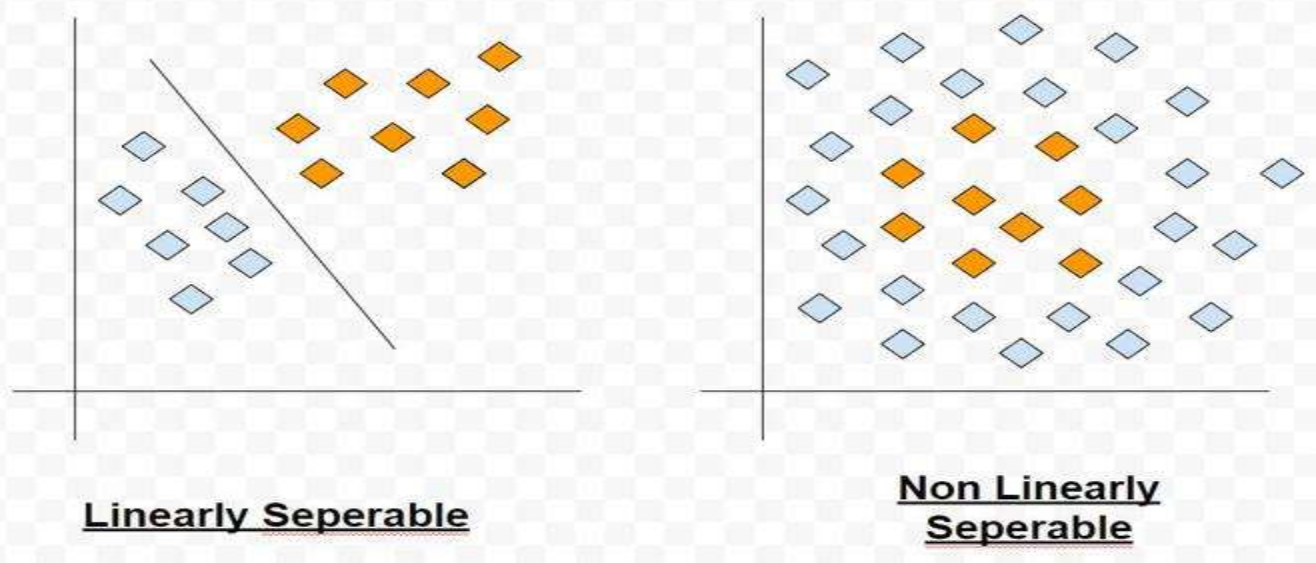
So, any point (x_1,x_2) which lies above the decision boundary, as depicted by the graph, will be assigned to class c1 and the points which lie below the boundary are assigned to class c2.

CLASSIFICATION PROBLEMS USING DECISION LINES (FOR 2-DIMENSIONAL SPACE)



CLASSIFICATION PROBLEMS USING DECISION LINES (FOR 2-DIMENSIONAL SPACE)

- For perceptron to work properly The two classes should be linearly separable i.e. the classes should be sufficiently separated from each other.



ADALINE NETWORK MODEL

- This model has only output neurons.
- The output value can be +1 or -1.
- A bias input x_0 (where x_0 =1) having a weight w_0 is added.
- The activation function is such that if weighted sum is positive or 0, the output is 1, else it is -1.

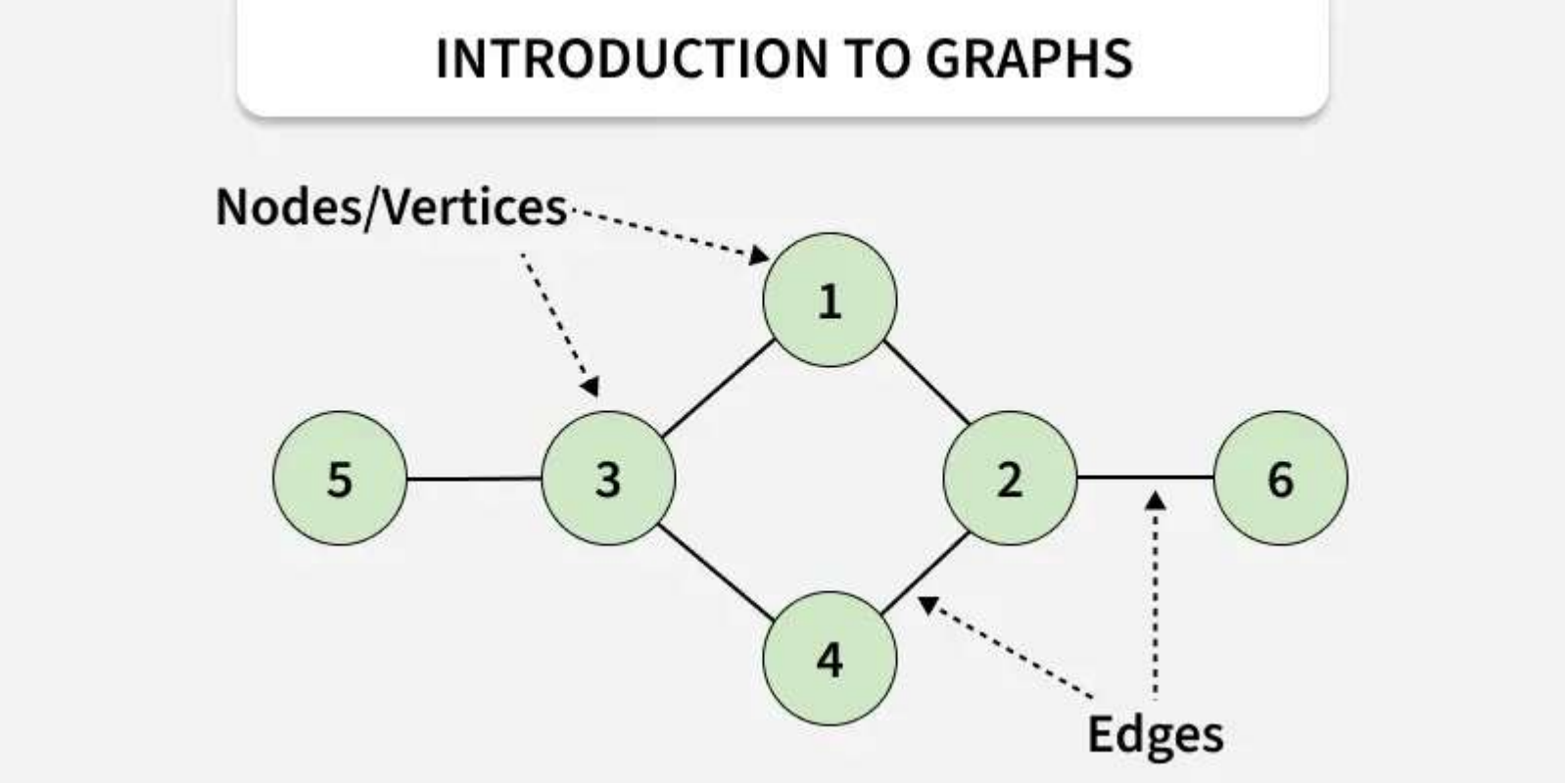
$$y_{sum} = \sum_{i=1}^{n} w_i x_i + b, where\, b = w_0$$

$$y_{out} = f(y_{sum}) = \begin{cases} 1, x \geq 1 \\ -1, x < 1 \end{cases}$$

A NEURAL NETWORK AS A DIRECTED GRAPH

- A neural network is represented as a directed graph where information flows in a specific direction from input to output.
- **Structural Components**
- **Nodes (Neurons):** These represent the vertices of the graph. Each node performs mathematical operations, such as calculating a weighted sum followed by an activation function.
- **Edges (Connections):** These represent the directed relationships between neurons, often called "links" or "arcs". They indicate the path along which data is transmitted.
- **Weights:** Numerical values assigned to edges that signify the strength or "cost" of the connection. During training, these weights are updated to optimize the network's performance

A NEURAL NETWORK AS A DIRECTED GRAPH



A NEURAL NETWORK AS A DIRECTED GRAPH

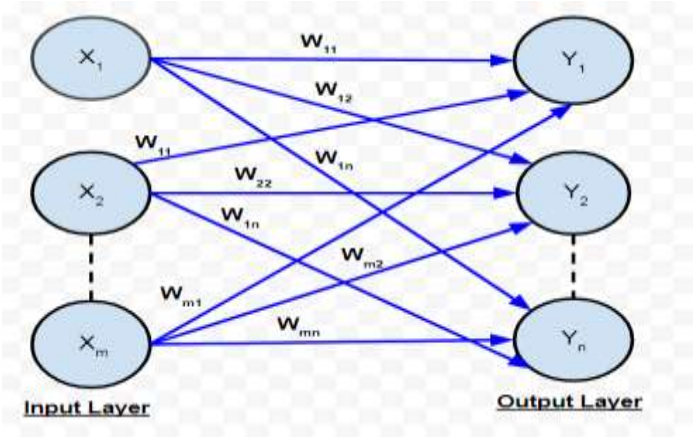
- Representation in Practice
- Computational Graphs: Neural networks use directed graphs to organize mathematical operations.
- A forward pass computes the output
- while a backward pass (backpropagation) uses the graph structure to calculate gradients.

ARCHITECTURES OF NEURAL NETWORK

- There are multiple possibilities for connecting the neurons based on which the **architecture** is defined.
- Single layer
- Multi layer
- Feed forward
- Recurrent
- Convulutional
- Radial basis function Network

ARCHITECTURES OF NEURAL NETWORK

- **Single-layer Feed Forward Network:**
- It consists of only two layers- the input layer and the output layer.
- **The input layer** consists of 'm' input neurons connected to each of the 'n' output neurons. The connections carry weights w_{11} and so on.

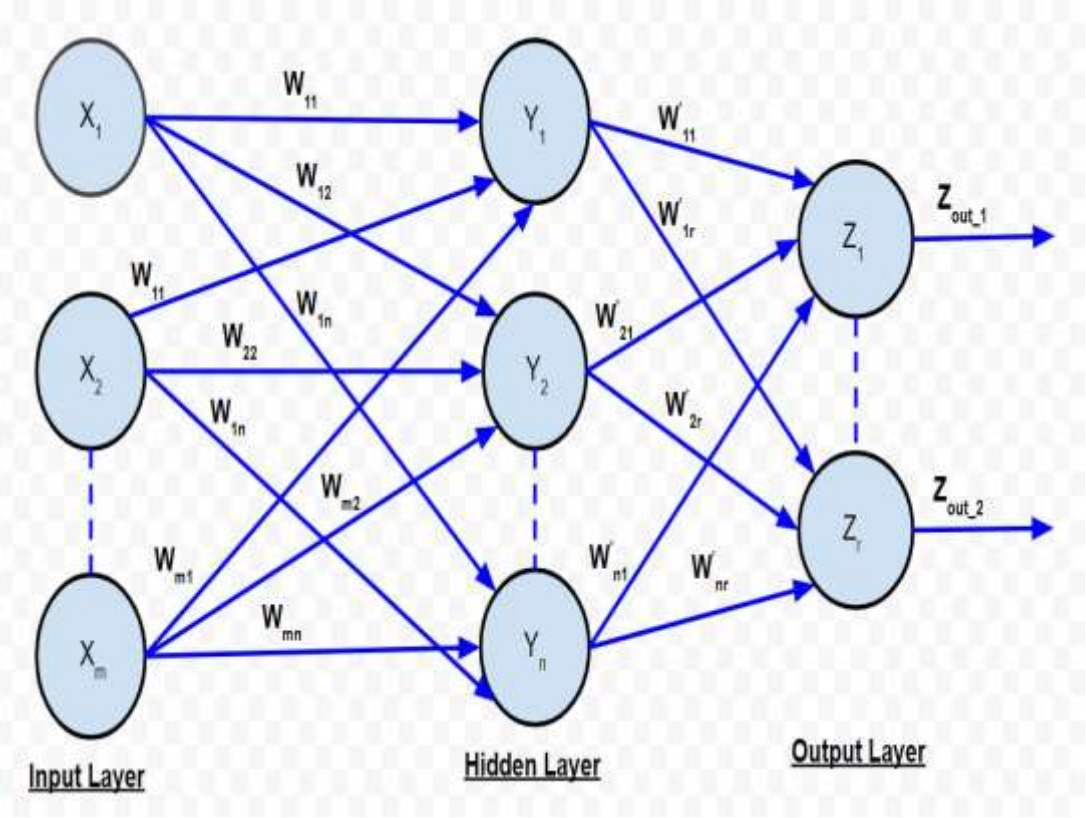


ARCHITECTURES OF NEURAL NETWORK

- The input layer of the neurons doesn't conduct any processing - they pass the i/p signals to the o/p neurons.
- The computations are performed in the output layer. So, though it has 2 layers of neurons, only one layer is performing the computation. This is the reason why **the network is known as SINGLE** layer.
- Also, the signals always flow from the input layer to the output layer. Hence, the **network is known as FEED FORWARD.**

ARCHITECTURES OF NEURAL NETWORK

Multi-layer Feed Forward Network:



The multi-layer feed-forward network is quite similar to the single-layer feed-forward network, except for the fact that there are one or more intermediate layers of neurons between the input and output layer. Hence, the network is termed as multi-layer.

Each of the layers may have a varying number of neurons.

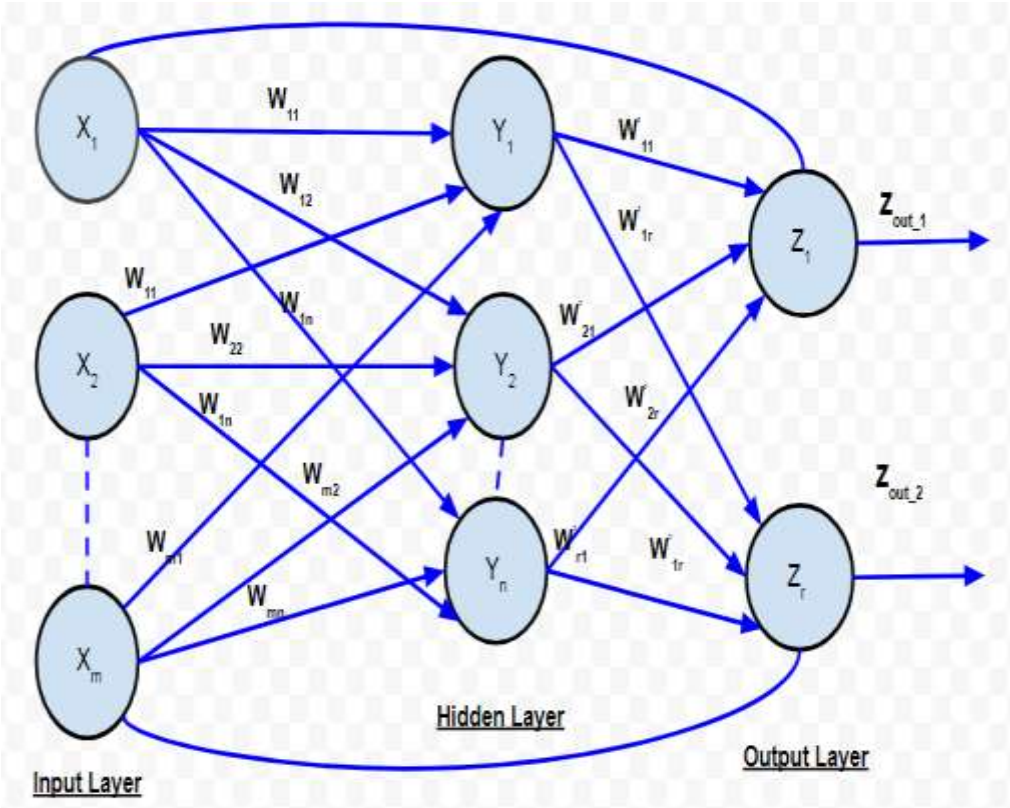
For example, the one shown in the diagram has 'm' neurons in the input layer and 'r' neurons in the output layer and there is only one hidden layer with 'n' neurons.

ARCHITECTURES OF NEURAL NETWORK

RECURRENT NETWORKS:

In feed-forward networks, the signal always flows from the input layer towards the output layer (in one direction only).

In the case of recurrent neural networks, there is a feedback loop (from the neurons in the output layer to the input layer neurons). There can be self-loops too.



ARCHITECTURES OF NEURAL NETWORK

Feedforward Networks: Data moves in a single direction through input, hidden, and output layers.

Convolutional Neural Network (CNN):

Convolutional Neural Networks (CNNs) are designed to process data that has a grid-like structure such as images.

- Apply filters to extract important features from the data such as edges or textures.
- image and speech recognition as they can identify patterns and structures in complex data.

ARCHITECTURES OF NEURAL NETWORK

Radial Basis Function Network (RBFN):

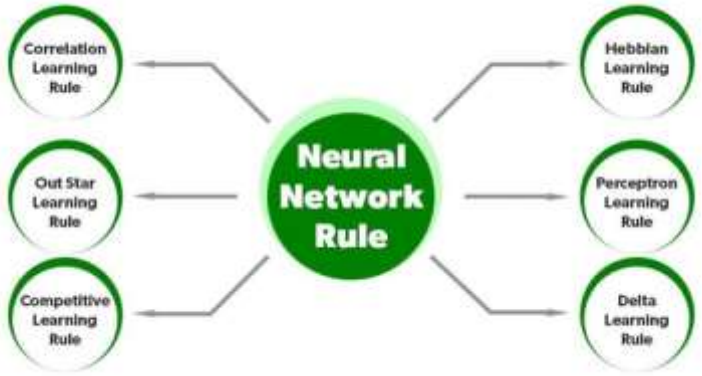
- Radial Basis Function Networks are designed to work with data that can be modeled in a radial or circular way.
- These networks consist of two layers: one that maps input to radial basis functions and another that finds the output.
- They are used for classification and regression tasks especially when the data represents an underlying pattern or trend.

Knowledge Representation (KR) in ANN

- Unlike traditional AI that uses rigid "if-then" rules, ANNs use a **distributed representation**.
- **Numerical Encoding:** Knowledge is not stored as discrete facts but is encoded in the **weights and biases** of the connections between neurons.
- **Embeddings:** Real-world entities (words, images, or objects) are converted into dense vectors (lists of numbers). Similar concepts are placed closer together in this mathematical space.
- **Latent Patterns:** Through training, the network identifies complex, non-linear correlations in data that are often invisible to human programmers.

LEARNING PROCESS:

- The learning rule enhances the Artificial Neural Network's performance by applying this rule over the network.
- Learning in ANNs involves adjusting the weights of the connections between neurons to improve performance on a given task.
- This adjustment is governed by learning rules, which define how weights are updated based on input data, output, and error.



LEARNING PROCESS

METHODS

1.HEBBIAN LEARNING

2.ERROR CORRECTION LEARNING(PERCEPTION,DELTA..)

3.BOLTZMAN LEARNING RULE

HEBBIAN LEARNING RULE

Principle: *This rule is based on the biological concept that "neurons that fire together, wire together.*

- Hebbian Learning Rule is an unsupervised learning algorithm used in neural networks to adjust the weights between nodes.
- Based on the principle that the connection strength between two neurons should change depending on their activity patterns.

HEBBIAN LEARNING RULE

The rule can be summarized as follows:

- When two neighboring neurons operate in the same phase at the same time, the weight between them increases.
- If the neurons operate in opposite phases, the weight between them decreases.
- When there is no signal correlation between the neurons, the weight remains unchanged.

HEBBIAN LEARNING RULE

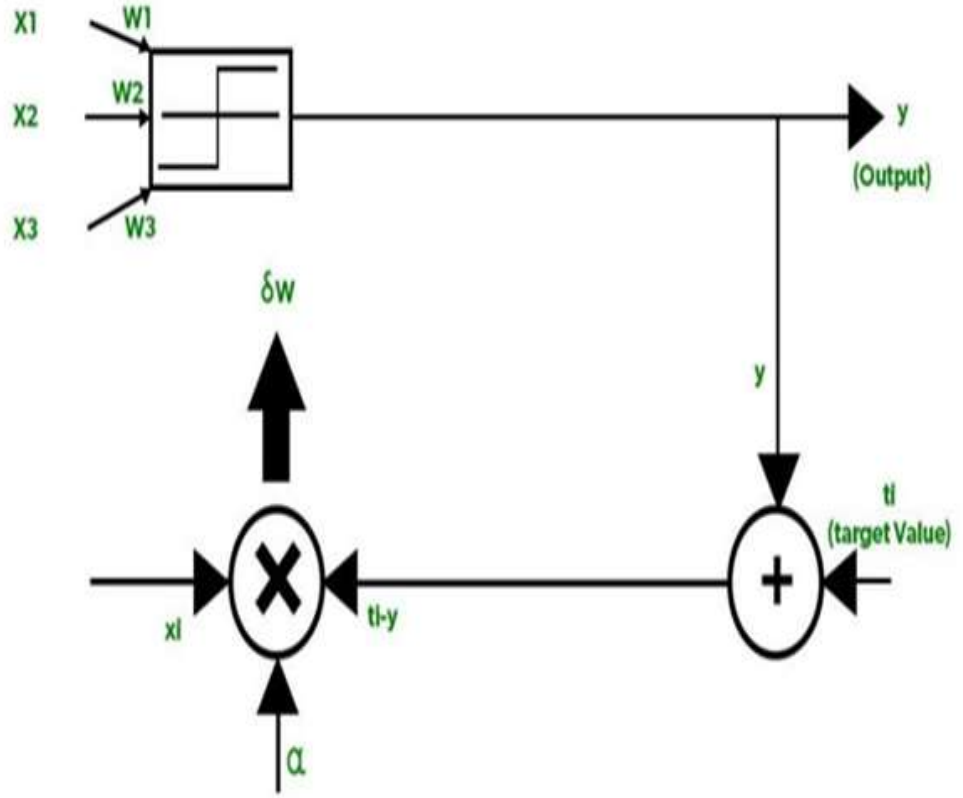
- The sign of the weight between two nodes is determined by the sign of their input signals:
- If both nodes receive inputs that are either positive or negative, the resulting weight is strongly positive.
- If one node’s input is positive while the other’s is negative, the resulting weight is strongly negative.

PERCEPTRON LEARNING RULE

- Principle: This rule adjusts weights to minimize classification errors.
- Perceptron Learning Rule is an error-correcting algorithm designed for single-layer feed forward networks.
- It is a supervised learning approach that adjusts weights based on the error calculated between the desired and actual outputs. Weight adjustments are made only when an error is present. The process is computed as follows:

PERCEPTRON LEARNING RULE

- The process is computed as follows:
  - $(x_1, x_2, x_3, \ldots, x_n)$: Set of input vectors
  - $(w_1, w_2, w_3, \ldots, w_n)$: Set of weights
  - y: Actual output
  - w_0: Initial weight
  - $w$new: New weight
  - δw: Change in weight
  - α: Learning rate



DELTA LEARNING RULE

- **Principle:** *Minimizes the error between the actual output and the desired output using gradient descent.*
- Delta Learning Rule is a supervised learning algorithm that uses a continuous activation function. Also known as the **Least Mean Square (LMS) method**,
- minimize the error across all training patterns.
- This rule is based on the **gradient descent approach**, which iteratively reduces error by updating the weights of the network.

BOLTZMANN LEARNING RULE

- *Inspired by statistical mechanics, this rule uses probabilities to adjust weights.*
- Boltzmann Learning Rule is a probabilistic learning method commonly used in stochastic networks such as Boltzmann Machines.
- It utilizes principles from statistical mechanics to find the optimal weight configuration that minimizes energy in the network.
- **Probabilistic Framework:** Learning is governed by the probability of a node's state, influenced by the system's energy.
- **Energy Minimization:** Weights are updated to minimize the network's energy, ensuring a stable and efficient configuration.

BOLTZMANN LEARNING RULE

- The change in weight (δw) is calculated using the correlation between the states of the input and output nodes:

$$\delta w = \alpha(P\text{data} - P\text{model})$$

Where:

- δw: Change in weight
- α: Learning rate
- $P\text{data}$: Probability of the data
- $P\text{model}$: Probability of the model's predictions

CREDIT ASSIGNMENT PROBLEM:

- When a neural network produces a result (whether correct or incorrect), the credit assignment problem asks: "Which parts of the network contributed to that result, and to what degree?"
- This is crucial for learning, as we need to know which connections to strengthen or weaken to improve performance.

CREDIT ASSIGNMENT PROBLEM:

- **Challenges**
- Hierarchical Networks:
- In deep neural networks with multiple layers, it's hard to trace the impact of a neuron in an early layer on the final output.
- The effect of a single neuron's activity is propagated through many layers, making it difficult to isolate its contribution.

CREDIT ASSIGNMENT PROBLEM:

Temporal Delays:

- In reinforcement learning, there can be a significant delay between an action and its resulting reward.
- The agent needs to figure out which of its past actions led to the reward, even if those actions occurred many steps earlier.

CREDIT ASSIGNMENT PROBLEM:

- **Solutions and Approaches:**

Backpropagation:

- This is the most widely used technique in supervised deep learning.
- It calculates the gradient of the error with respect to each weight in the network, allowing us to determine how much each weight contributed to the error.

Reinforcement Learning Techniques:

- Temporal Difference (TD) learning:
- These methods learn from predictions and updates based on the difference between predicted and actual rewards.

CREDIT ASSIGNMENT PROBLEM:

Monte Carlo methods:

- These methods learn from complete episodes of experience, assigning credit to actions based on the total reward received.

Eligibility traces:

- These techniques keep track of the recent history of actions, allowing the agent to assign credit to actions that occurred earlier in a sequence.

STATISTICAL NATURE OF THE LEARNING PROCESS IN NEURAL NETWORKS

- Understanding the statistical nature of the learning process in neural networks (NNs) is pivotal for optimizing their performance.
- consider the following case
- Let the true function be F(x,T) and the model approximation be F(x,w).
- The total prediction error can be thought of as coming from two main sources:
- **Bias B(w)**
  - Error due to overly simple assumptions in the model.
  - High bias ⇒ model is too simple ⇒ underfitting.
  - Example: Using a straight line to fit a highly curved relationship.

STATISTICAL NATURE OF THE LEARNING PROCESS IN NEURAL NETWORKS

- **Variance V(w)**
  - Error due to sensitivity to training data.
  - High variance ⇒ model is too complex ⇒ over fitting.
- For good performance:
- Bias ↓ and Variance↓
- But in practice:
- Reducing bias usually increases variance.
- Reducing variance usually increases bias.
  This conflict is called the **bias–variance trade-off** or dilemma.

STATISTICAL NATURE OF THE LEARNING PROCESS IN NEURAL NETWORKS

- By knowing above
- Design models that balance both, and understanding statistical nature helps us Using problem-specific architectures and constraints.

Thank You..