GLA UNIVERSITY
MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A+ Grade by NAAC
12-B Status from UGC

BCA III Year VII Semester

BCAC 0028 SOFT COMPUTING

Module 1

Introduction To Neural Networks: Neural Networks Neuron, Nerve Structure And Synapse, Artificial Neuron And Its Model, Activation Functions.

Neural Network Architecture: Single Layer And Multilayer Feed Forward Networks, Recurrent Networks. Perception And Convergence Rule. Supervised Learning Network & Unsupervised Learning Network.

Contd…

Back Propagation Networks-I: Perceptron Model, Solution, Single Layer, Multilayer Perception Model.

Back Propagation Networks-II: Back Propagation Learning Methods, Effect Of Learning Rule Co-Efficient; Back Propagation Algorithm, Applications.

Concept of Computing

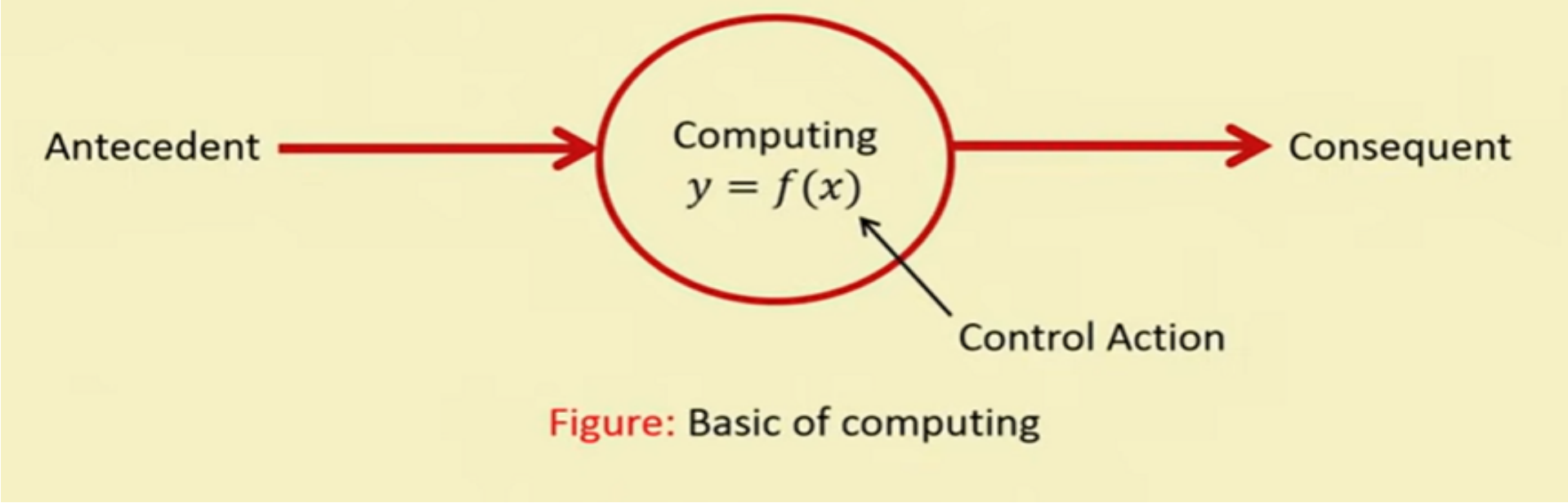


Figure: Basic of computing

Figure : Basic of computing

$y = f(x)$, f is a mapping function
f is also called a formal method or an algorithm to solve a problem.

Concept of Computing

Antecedent: a thing or an event that exists or comes before another, and may have influenced it.

Consequent: following as the result of something else.

Important Characteristics of Computing

1. Should provide precise solution.

1. Control action should be unambiguous and accurate.

1. Suitable for problem, which is easy to model mathematically

Hard Computing

In 1996, LA Zade (LAZ) introduced the term hard computing.

According to LAZ: We term a computing as ”Hard” computing, if

1. Precise result is guaranteed
2. Control action is unambiguous
3. Control action is formally defined (i.e. with mathematical model or algorithm)

Hard Computing

Example of Hard Computing:

Solving numerical problems (e.g. Roots of polynomials, Integration etc.)

Searching and sorting techniques

Solving ”Computational Geometry” problems (e.g. Shortest tour in Graph theory, Finding closest pair of points etc.)

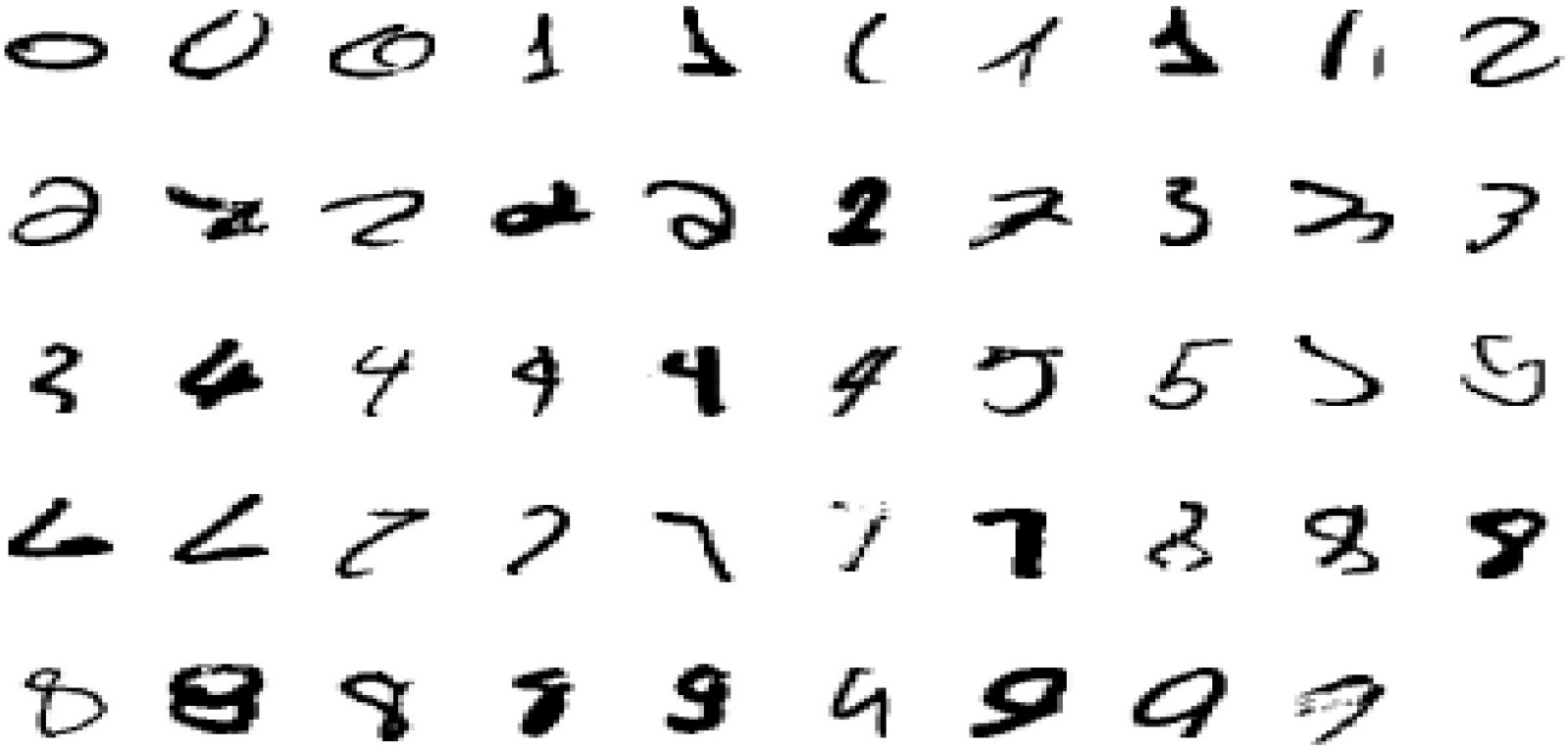
Problem Areas of Hard Computing

Problems in some other areas of applications

- Medical diagnosis
- Person identification
- Computer vision
- Hand written character recognition
- Pattern recognition and Machine Intelligence MI
- Weather forecasting

GLA UNIVERSITY MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A+ Grade by NAAC
12-B Status from UGC

A classic example of a task that requires machine learning: It is very hard to say what makes a 2



1. **Recognizing patterns:**
   - Facial identities or facial expressions
   - Handwritten or spoken words
   - Medical images
2. **Recognizing anomalies:**
   - Unusual sequences of credit card transactions
   - Unusual patterns of sensor readings in a nuclear power plant or unusual sound in your car engine.

Some more Web based examples….

1. **The web contains a lot of data. Tasks with very big datasets often use machine learning**
   - especially if the data is noisy or non-stationary.
2. **Spam filtering, fraud detection:**
   - The enemy adapts so we must adapt too.
3. **Recommendation systems:**
   - Lots of noisy data. Million dollar prize!
4. **Information retrieval:**
   - Find documents or images with similar content.

- The term soft computing was proposed by the inventor of fuzzy logic, Lofti A. Zadeh. He describes it as follows.

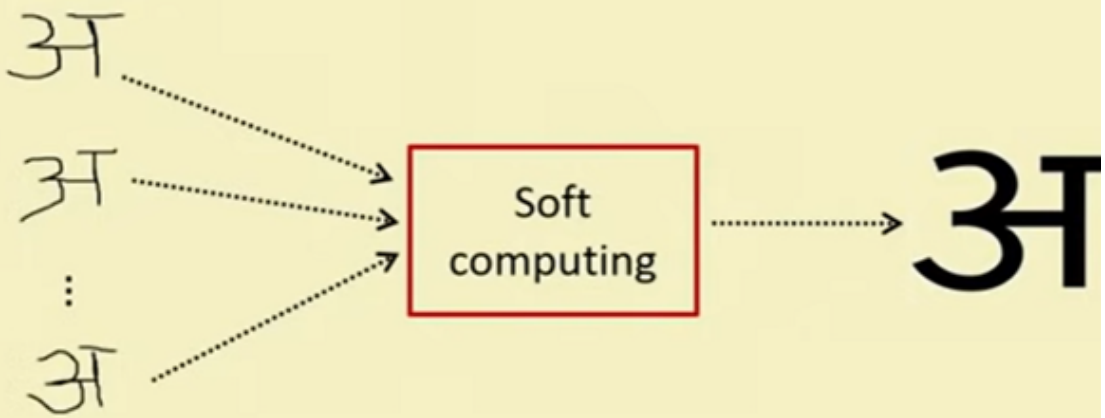
- **Definition**: Soft computing is a collection of methodologies that aim to exploit the tolerance for imprecision and uncertainty to achieve tractability, robustness, and low solution cost, Its principal constituents are fuzzy logic ,neuro-computing, and probabilistic reasoning.
- The role model for soft computing is the human mind.

Characteristics of Soft Computing

1. It does not require any mathematical modeling of problem solving.
2. It may not yield the precise solution
3. Algorithms are adaptive (i.e. it can adjust to the change of dynamic environment)
4. Use some biological inspired methodologies such as genetics, evolution, Ant's behaviors, particles swarming, human nervous systems etc.

GLA UNIVERSITY MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A+ Grade by NAAC
12-B Status from UGC

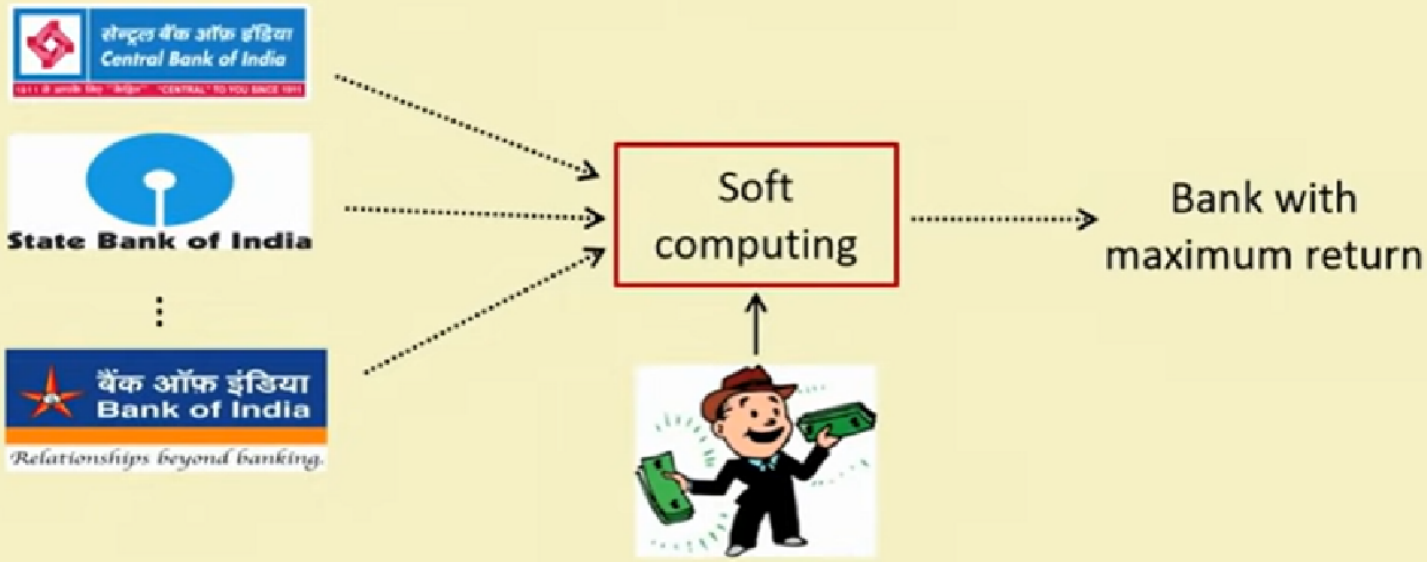
Examples of soft computing



Example: Hand written character recognition
(Artificial Neural Networks)

GLA UNIVERSITY MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A+ Grade by NAAC
12-B Status from UGC

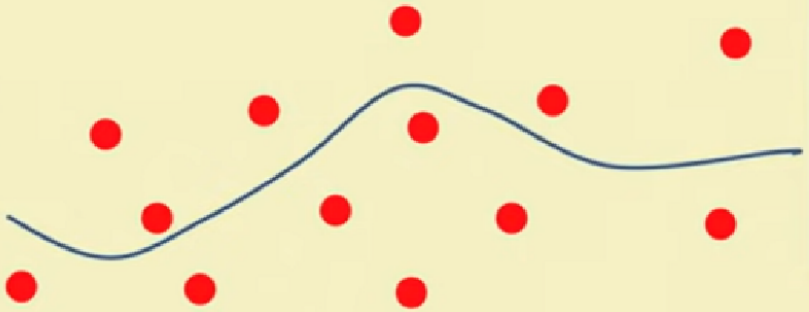
Examples of soft computing



Example: Money allocation problem
(Evolutionary Computing)

GLA UNIVERSITY MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A+ Grade by NAAC
12-B Status from UGC

Examples of soft computing



Example: Robot movement
(Fuzzy Logic)

How Soft Computing?

How a student learns from his teacher?

1. Teacher asks questions and tell the answers then.
2. Teacher puts questions and hints answers and asks whether the answers are correct or not.
3. Student thus learn a topic and store in his memory.
4. Based on the knowledge he solves new problems.
5. This is the way how human brain works.
6. Based on this concepts Artificial Neural Networks is used to solve problem

Hard Computing vs Soft Computing

Hard Computing	Soft Computing
It requires a precisely stated analytical model and often a lot of Computation time	It is tolerant of imprecision, uncertainty, partial truth and approximation
It is based on binary logic crisp system, numerical analysis and crisp software.	It is based on fuzzy logic, neural nets and probabilities reasoning.
It has the characteristics of precision and categoricity.	It has the characteristics of approximation and dispositionality

Hard Computing vs Soft Computing

Hard Computing	Soft Computing
It is deterministic	It incorporates stochasticity
It requires exact input data	It can deal with ambiguous and noisy data
It is strictly sequential.	It allows parallel computations
It produces precise answers	It can yield approximate answers

Hybrid Computing

Hybrid Computing

It is a combination of the conventional hard computing and emerging soft computing

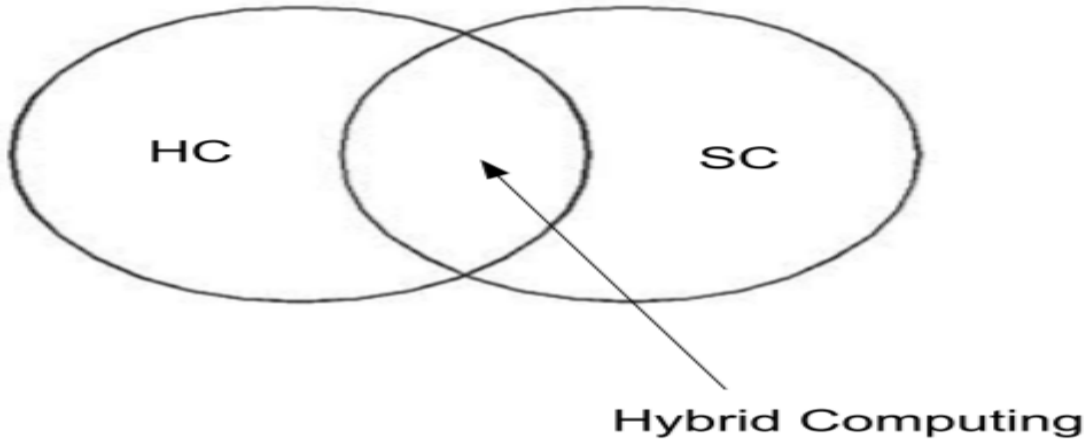


Figure : Concept of Hybrid Computing

Questions

Q1. Differentiate Hard Computing and Soft Computing?

Q2. What are the problems which can be solved by soft computing?

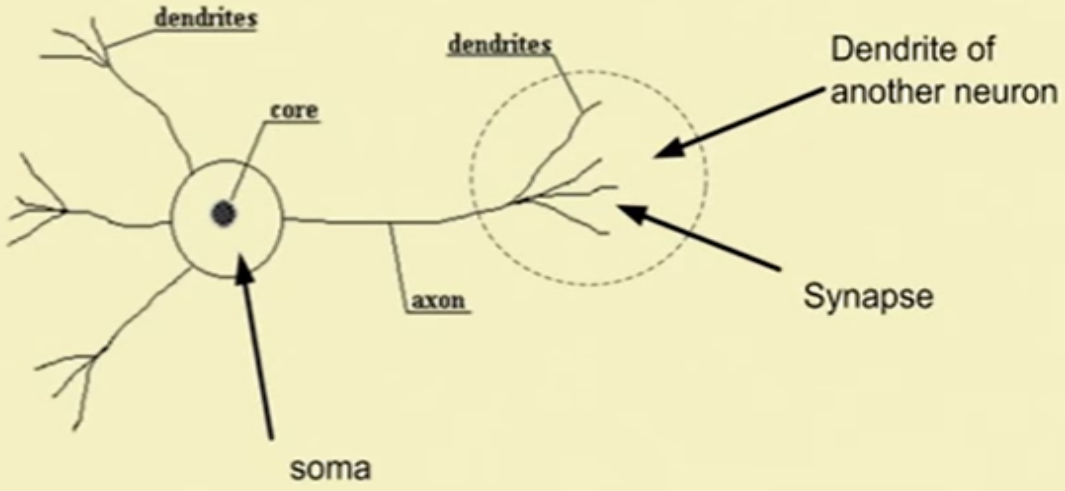
GLA UNIVERSITY MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A+ Grade by NAAC
12-B Status from UGC

Biological nervous system

Neuron and its working

Figure shows a schematic of a biological neuron. There are different parts in it

- **Dendrite** : A bush of very thin fibre.
- **Axon** : A long cylindrical fibre.
- **Soma** : It is also called a cell body, and just like as a nucleus of cell.
- **Synapse** : It is a junction where axon makes contact with the dendrites of neighbouring dendrites.



Working of Neuron

A neuron is a special cell in our brain and body that helps send and receive messages. Think of it like a tiny messenger. **How It Works (Simple Steps):**

Message Comes In-

Other neurons or body parts send a signal (like a message) to the dendrites (the receiving parts of a neuron).

Message Travels-

The signal travels through the cell body (where the neuron decides what to do with the message).

Contd…

Message Goes Out

If the message is strong enough, it travels quickly down a long part called the axon.

Message Reaches the End

At the end of the axon are axon terminals, which pass the message to the next neuron or muscle using tiny chemicals called neurotransmitters.

Contd…

Analogy- Like a WhatsApp Message
Dendrites = Phone receiving a message
Cell body = You read and think about it
Axon = You decide to reply and type
Axon terminal = You hit send
Neurotransmitters = The message is delivered to your friend.

Question- ***Various types of Neurons?***

Artificial Neural Networks (ANNs)

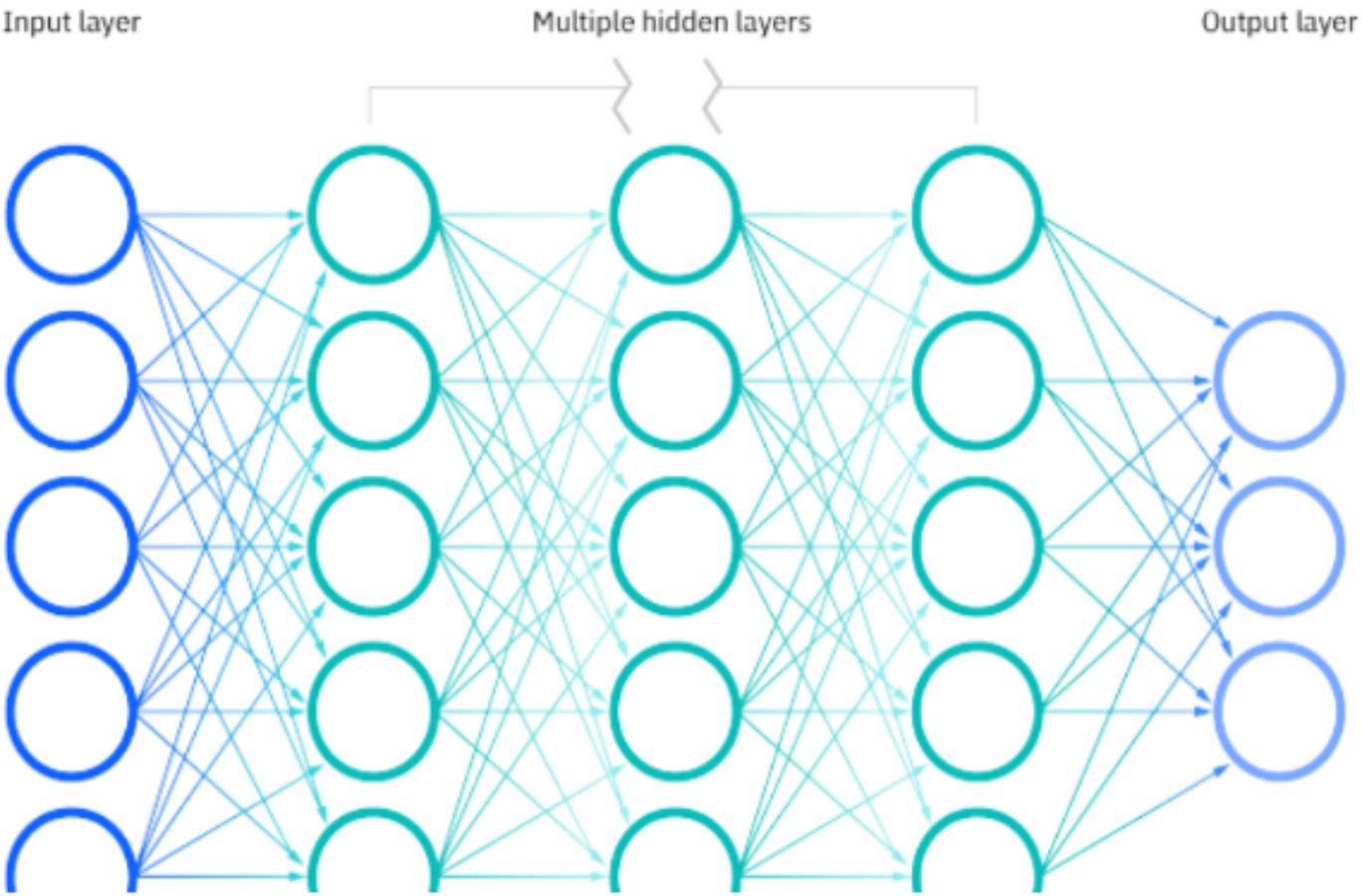
- Neural networks are also artificial neural networks (ANNs) or simulated neural networks (SNNs).
- It is a subset of machine learning at the heart of deep learning algorithms.
- Their name and structure are inspired by the human brain, mimicking the way that biological neurons signal to one another.

Artificial Neural Networks (ANNs)

- Artificial neural networks (ANNs) are comprised of a node layers, containing an input layer, one or more hidden layers, and an output layer.
- Each node, or artificial neuron, connects to another and has an associated weight and threshold.
- If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network.
- Otherwise, no data is passed along to the next layer of the network.

GLA UNIVERSITY MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A+ Grade by NAAC
12-B Status from UGC

Elements Artificial Neural Networks



Elements of a Neural Network

- **Input Layer:** This layer accepts input features. It provides information from the outside world to the network, no computation is performed at this layer, nodes here just pass on the information(features) to the hidden layer.
- **Hidden Layer:** Nodes of this layer are not exposed to the outer world, they are part of the abstraction provided by any neural network. The hidden layer performs all sorts of computation on the features entered through the input layer and transfers the result to the output layer.
- **Output Layer:** This layer bring up the information learned by the network to the outer world.

Neuron Components

(i) A set of **'i' synapses having weight wi.** A signal xi forms the input to the i-th synapse having weight wi. The value of any weight may be positive or negative. A positive weight has an extraordinary effect, while a negative weight has an inhibitory effect on the output of the summation junction.

(ii) A **summation junction** for the input signals is weighted by the respective synaptic weight. Because it is a linear combiner or adder of the weighted input signals, the output of the summation junction can be expressed as follows:

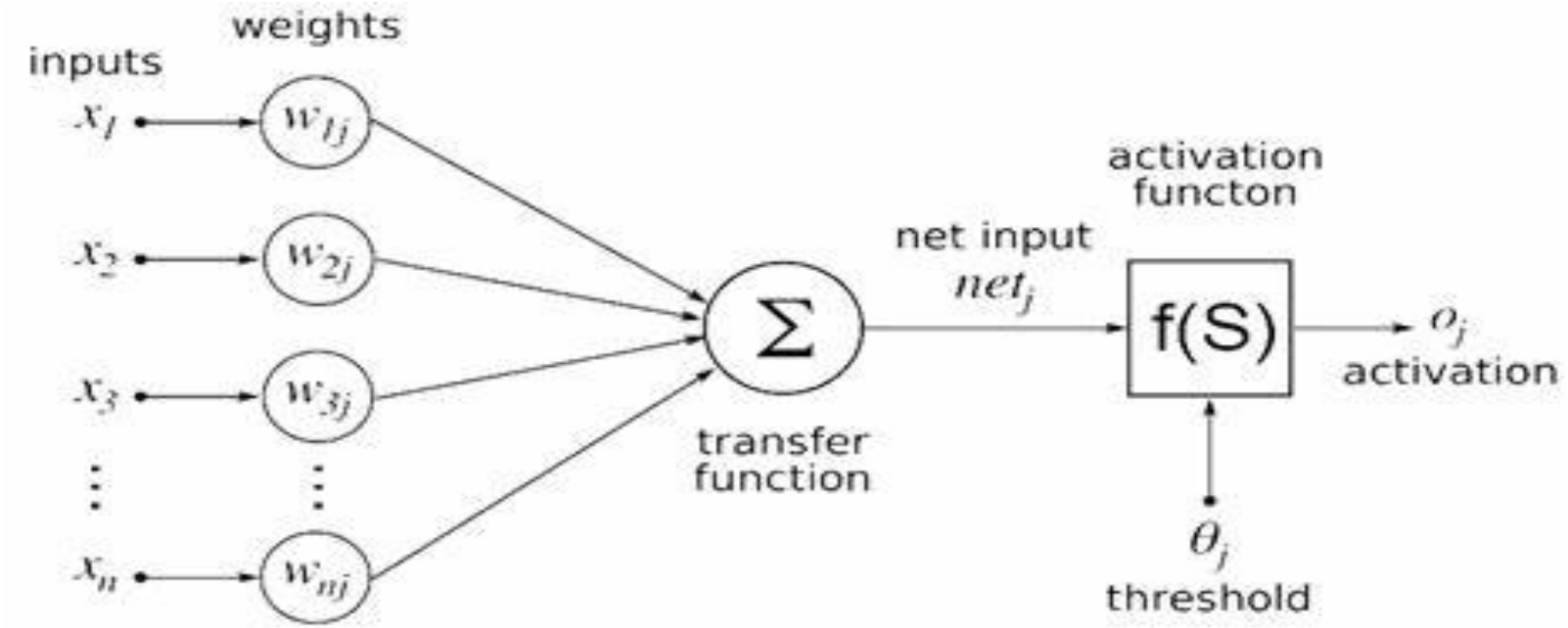
$$y_{sum} = \sum_{i=1}^{n} w_i x_i$$

Neuron Components

(iii) A threshold **activation function** (or simply the **activation function**, also known as a **squashing function**) results in an output signal only when an input signal exceeding a specific threshold value comes as an input. It is similar in behaviour to the biological neuron, which transmits the signal only when the total input signal meets the firing threshold.

** Threshold Value:- Reflects the minimum performance required to achieve the required operational effect while being achievable through the current state of technology at an affordable life-cycle cost.

GLA UNIVERSITY
MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A+ Grade by NAAC
12-B Status from UGC



Artificial neural network

- Here, $x_1, x_2, \cdots, x_n$ are the n inputs to the artificial neuron. $w_1, w_2, \cdots, w_n$ are weights attached to the input links.
- Note that, a biological neuron receives all inputs through the dendrites, sums them and produces an output if the sum is greater than a threshold value.
- Hence, the total input say I received by the soma of the artificial neuron is $I = w_1x_1 + w_2x_2 + \cdots + w_nx_n = \sum^n_{i=1} w_ix_i$
- To generate the final output y, the sum is passed to a filter ϕ called transfer function, which releases the output. That is, $y = \phi(I)$.

Mathematical Understanding

- A very commonly known transfer function is the **thresholding function**.
- In this thresholding function, sum (i.e. I) is compared with a threshold value θ.
- If the value of I is greater than θ, then the output is 1 else it is 0 (this is just like a simple linear filter).
- In other words, $y = \phi(\sum^{n}_{i=1} w_i x_i - \theta)$

  where $\phi(I) = 1$, if $I > \theta$

  0 , if $I \leq \theta$ Such a Φ is called **step function** (also known as Heaviside function).

Types of Activation Function

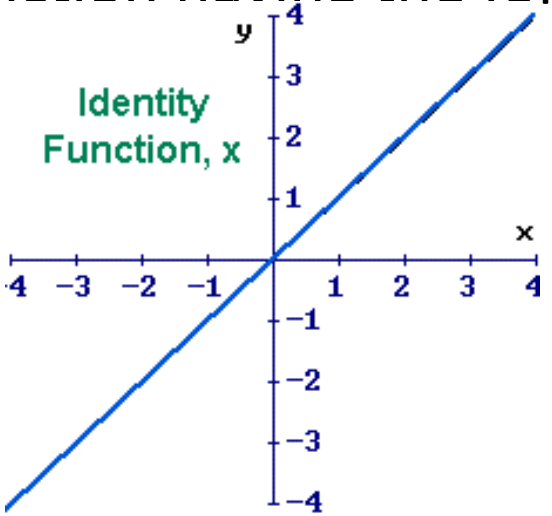
The most commonly used activation function are listed below:

1. Identity Function
2. Threshold/step Function
3. Signum Function
4. ReLU (Rectified Linear Unit) Function
5. Sigmoid Function
6. Hyperbolic Tangent Function

Types of Activation Function

It's just a thing function that you use to get the output of node. It is also known as ***Transfer Function****.*

(i) **Identity Function:** The identity function is used as an activation function for the input layer. The identity function is a special case of an activation function where the output signal is equal to the input signal. In other words, the identity function simply passes the input signal through unchanged. It is a linear function having the form:

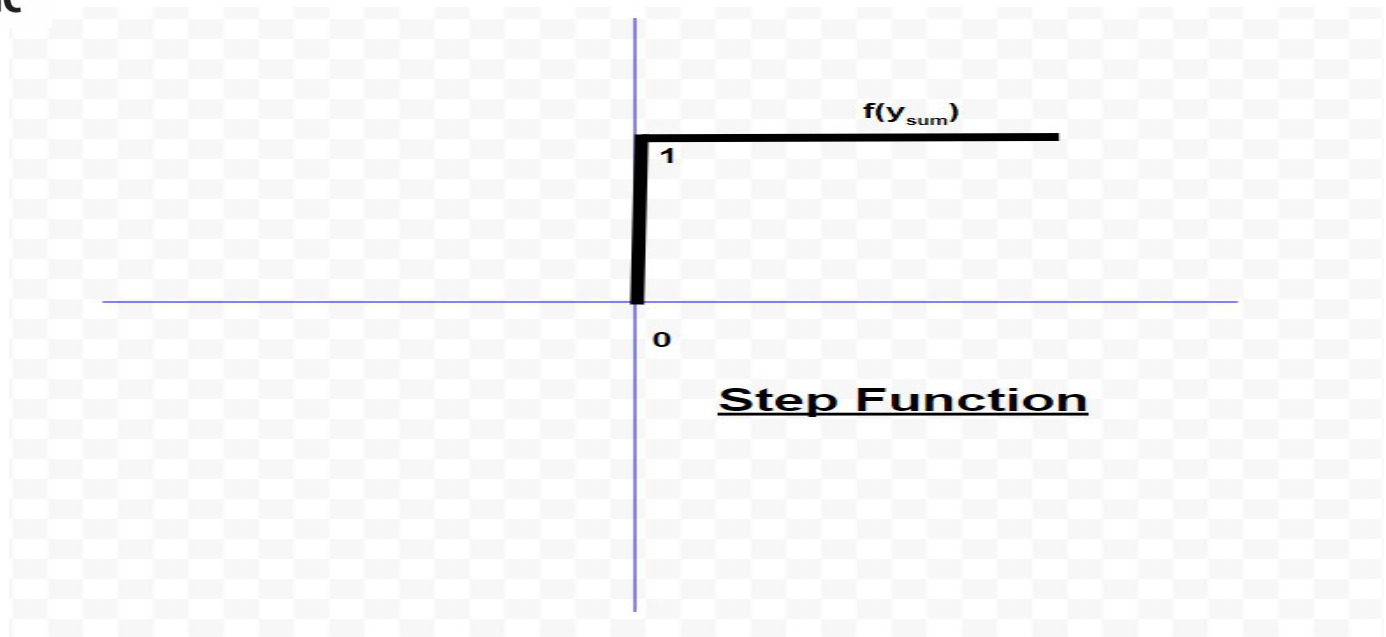


GLA UNIVERSITY MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A+ Grade by NAAC
12-B Status from UGC

Types of Activation Function

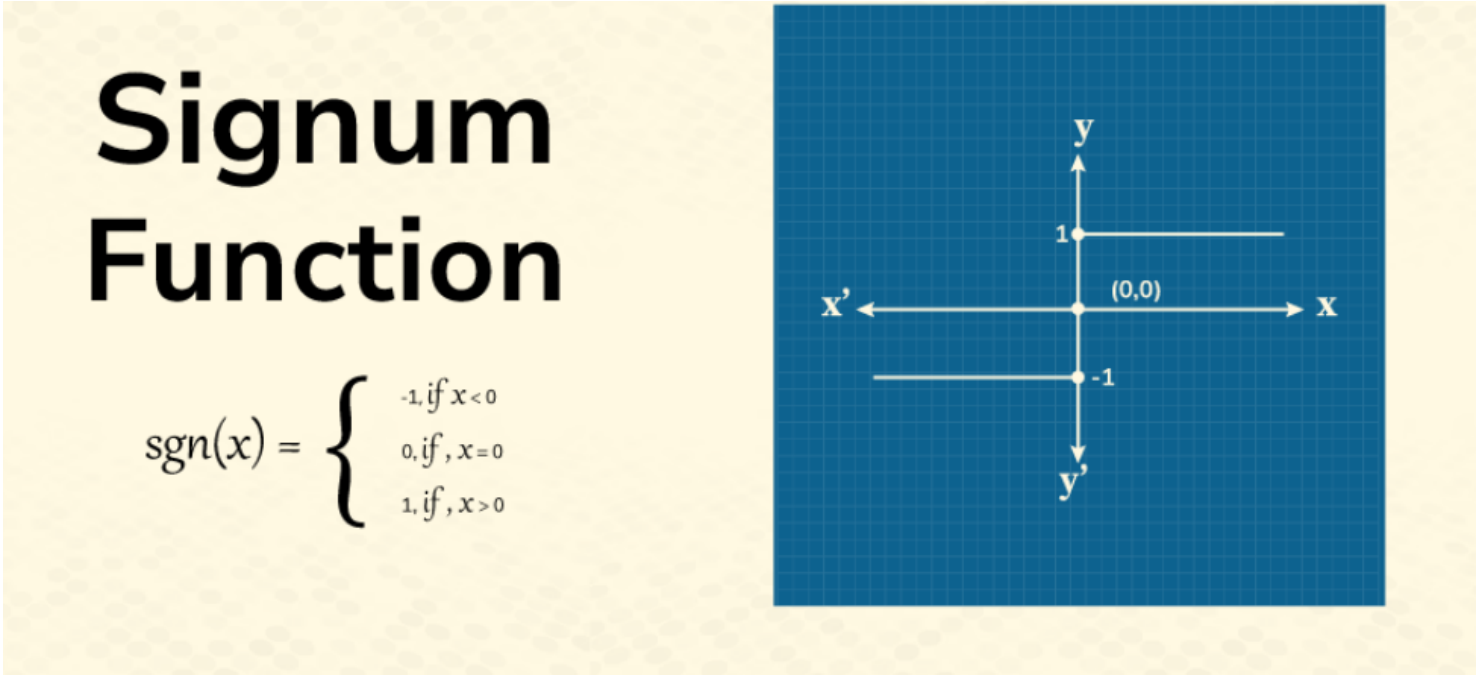
(ii) **Threshold/step Function:** It is a commonly used activation function. As depicted in the diagram, it gives **1 as an output** of the input is either 0 or positive. If the input is negative, it gives **0 as output**. Example is Fan on/off.

Types of Activation Function



$$y_{out} = f(y_{sum}) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases}$$

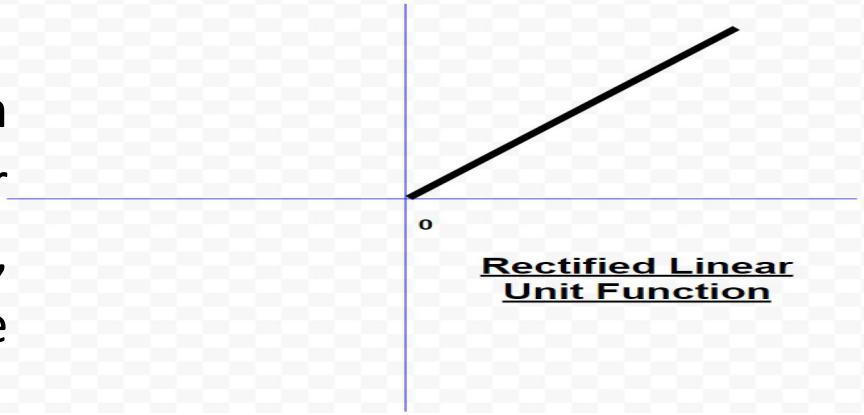
iii) **Signum Function**-Imagine a temperature sensor that only tells you if it's cold, hot, or perfectly neutral — not the exact temperature.



Types of Activation Function

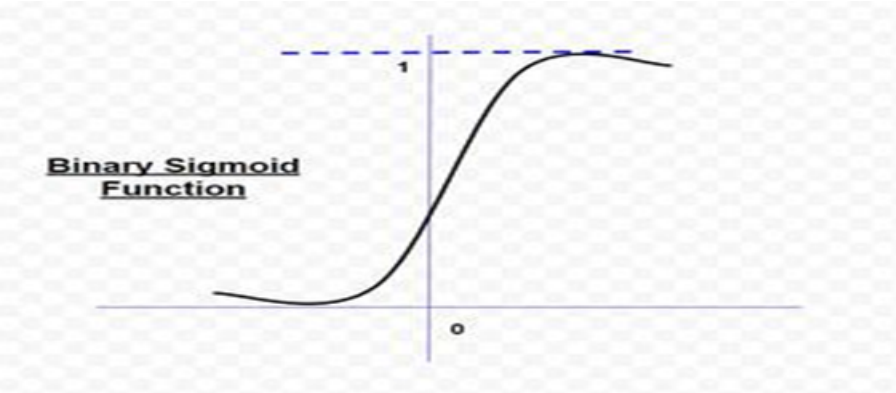
(iv) **ReLU (Rectified Linear Unit) Function:** It is the most popularly used activation function in the areas of convolutional neural networks and deep learning. Example -the fan is off until it gets hot. Then, speed increases with temperature.

This means that f(x) is zero when x is less than zero and f(x) is equal to x when x is above or equal to zero. **This function is differentiable**, except at a single point x = 0. In that sense, the derivative of a ReLU is actually a sub-derivative.



$$f(x) = \begin{cases} x, x \geq 0 \\ 0, x < 0 \end{cases}$$

(iv)



$$y_{out} = f(x) = \frac{1}{1+e^{-kx}}$$

GLA UNIVERSITY MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A+ Grade by NAAC
12-B Status from UGC

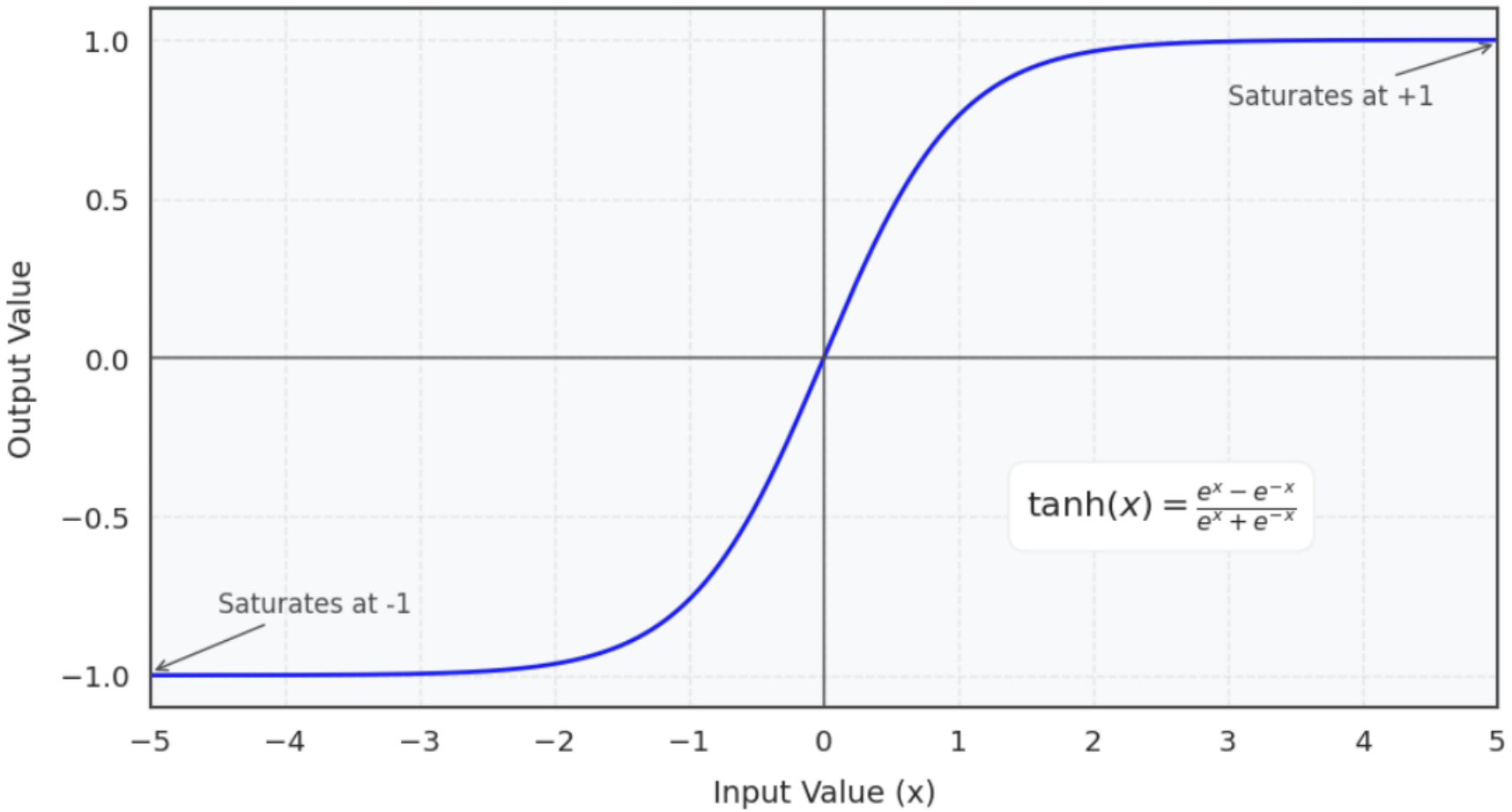
(vi) **Hyperbolic Tangent Function**

- It is bipolar in nature. It is a widely adopted activation function for a special type of neural network known as **Backpropagation Network.** The hyperbolic tangent function is of the form
- This function is similar to the bipolar sigmoid function.
- Like Sigmoid, but fan can blow cold air too (for cold input).

GLA UNIVERSITY
MATHURA
Recognised by UGC Under Section 2(f)

Accredited with A+ Grade by NAAC

12-B Status from UGC



Step	Light switch (ON/OFF)
Sigmoid	Fan slowly turns on
ReLU	Fan only works when hot
Tanh	Fan can blow cold or hot air
Signum	Just tells direction: forward, stop, back

Why Are Activation Functions Necessary?

Neural networks consist of multiple layers where neurons process input signals and pass them to subsequent layers.

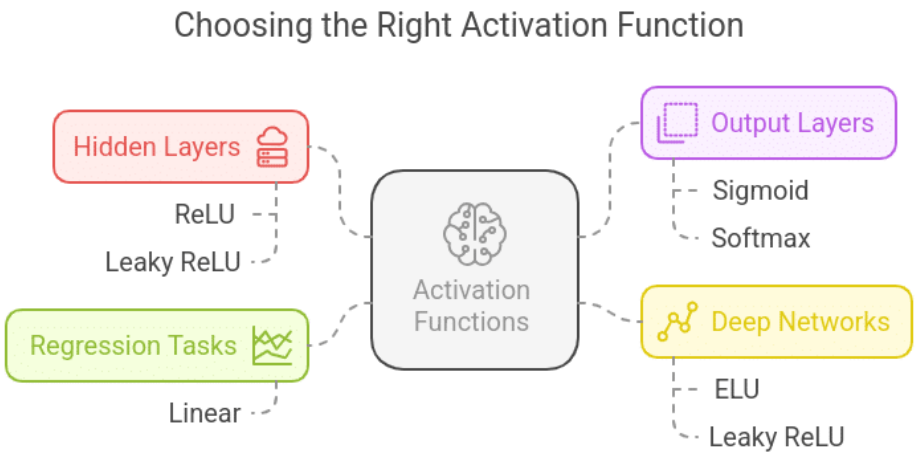
Everything inside a neural network becomes a basic linear transformation when activation functions are removed, which renders the network unable to discover complex features.

Key reasons why activation functions are necessary:

- **Introduce non-linearity**: Real-world problems often involve complex, non-linear relationships. Activation functions enable neural networks to model these relationships.
- **Enable hierarchical feature learning**: Deep networks extract multiple levels of features from raw data, making them more powerful for **Pattern Recognition**.
- **Prevent network collapse**: Without activation functions, every layer would perform just a weighted sum, reducing the depth of the network into a single linear model.
- **Improve convergence during training**: Certain activation functions help improve gradient flow, ensuring faster and more stable learning.

How to Choose the Right Activation Function?

- **For hidden layers,** ReLU or Leaky ReLU is recommended due to efficient gradient propagation.
- **For binary classification,** Sigmoid is commonly used in the output layer.
- **For Multiclass classification problem,** Softmax is the preferred choice.
- **For deep networks:** Consider using ELU or Leaky ReLU to avoid dead neurons.
- **For regression tasks,** Linear activation is used when predicting continuous values.



Real-World Applications

Neural networks require activation functions as their main component to transform data into complex predictions that achieve accuracy. The applications of activation functions exist throughout real-world scenarios in the following implementations:

1. Image Recognition (ReLU & Softmax in CNNs)

Example: Face Recognition in Smartphones

- In facial recognition systems like Face ID, CNNs analyze facial features using ReLU activation in hidden layers to process pixel values efficiently.
- The final layer uses Softmax activation, which assigns probability scores to different faces to identify the correct person.

2. Natural Language Processing (NLP) (Tanh & Softmax in LSTMs & Transformers)

Example: Chatbots & Virtual Assistants

- Virtual assistants like Alexa, Siri, and Google Assistant use deep learning models with Tanh activation in LSTMs to understand sentence context.
- The last layer of language models uses Softmax activation to predict the most probable next word or response.

Real-World Applications

3. Healthcare – Medical Diagnosis (ReLU & Sigmoid in CNNs & DNNs)

Example: Cancer Detection from Medical Images

- In medical imaging (e.g., detecting tumors from MRI scans), **CNNs with ReLU activation** help extract important image features.
- The final layer uses **Sigmoid activation** to classify an image as either "cancerous" (1) or "non-cancerous" (0), helping doctors with early diagnosis.

4. **Autonomous Vehicles (ReLU & Leaky ReLU in Deep Reinforcement Learning)**

Example: Self-Driving Cars (Tesla Autopilot)

- Self-driving cars process real-time sensor data using deep learning.
- **ReLU activation** in neural networks helps recognize objects like pedestrians, road signs, and vehicles.
- **Leaky ReLU activation** prevents inactive neurons from making split-second driving decisions.

1. **Fraud Detection (Sigmoid in Binary Classification Models)**

- **Example:** Credit Card Fraud Detection
- Banks use AI to detect fraudulent transactions by analyzing patterns in spending behavior.
- A **binary classification neural network** uses **Sigmoid activation** to classify whether a transaction is "fraud" (1) or "legitimate" (0).

Numerical Problems

1. Identity Activation Function
    1. Inputs: $[x_1, x_2, x_3] = [0.5, 0.7, 0.3]$, Weights: $[w_1, w_2, w_3] = [0.2, 0.4, 0.1]$, Bias: $b = 0.1$
    2. Inputs: [0.5, 0.7], Weights: [0.4, 0.6], Bias = 0.2
    3. Inputs: [0.9, 0.4, 0.3], Weights: [0.2, 0.5, 0.1], Bias = 0.0
    4. Inputs: [1.0, 2.0, 3.0], Weights: [0.5, 0.25, 0.1], Bias = 1.0
2. Binary Step Function
3. Bipolar Step Function
    1. Inputs: [0.6,0.4], Weights=[0.5, 0.5], Bias = -0.5
    2. Inputs: [0.2,0.3, 0.1], Weights=[0.6, 0.6, 0.6], Bias = -0.5
    3. Inputs: [1.0,1.0], Weights=[0.7, 0.7], Bias = -1.2
4. Binary Sigmoid
5. Bipolar Sigmoid
6. Hyperbolic Tangent(Tanh)
7. Rectified Linear Unit(ReLU)

Numericals

1.

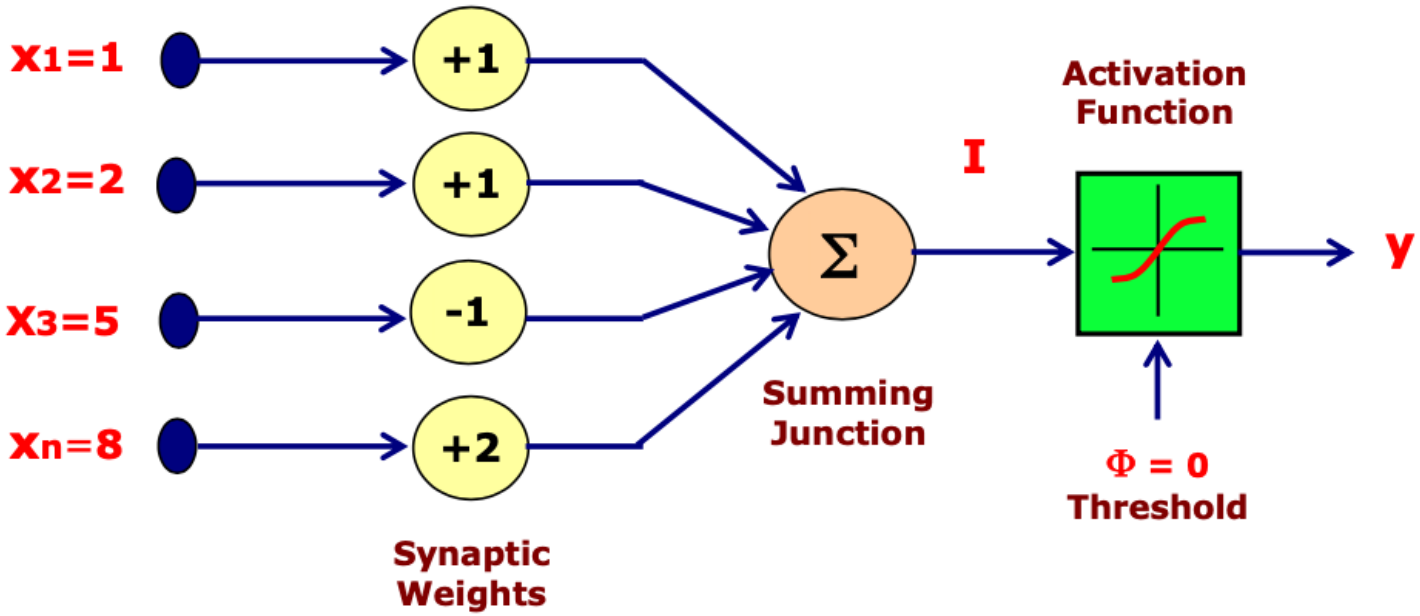


Fig Neuron Structure of Example

Numericals
2.

Inputs:

$x1=2.5, x2=1.2, x3=-0.5, x4=3.0$

Weights:

$w1=0.8, w2=-1.5, w3=0.3, w4=1.1$

Apply Sigmoid Function.

Numericals
3.

Inputs:

$x1=-1.0, x2=0.5, x3=2.3$

Weights:

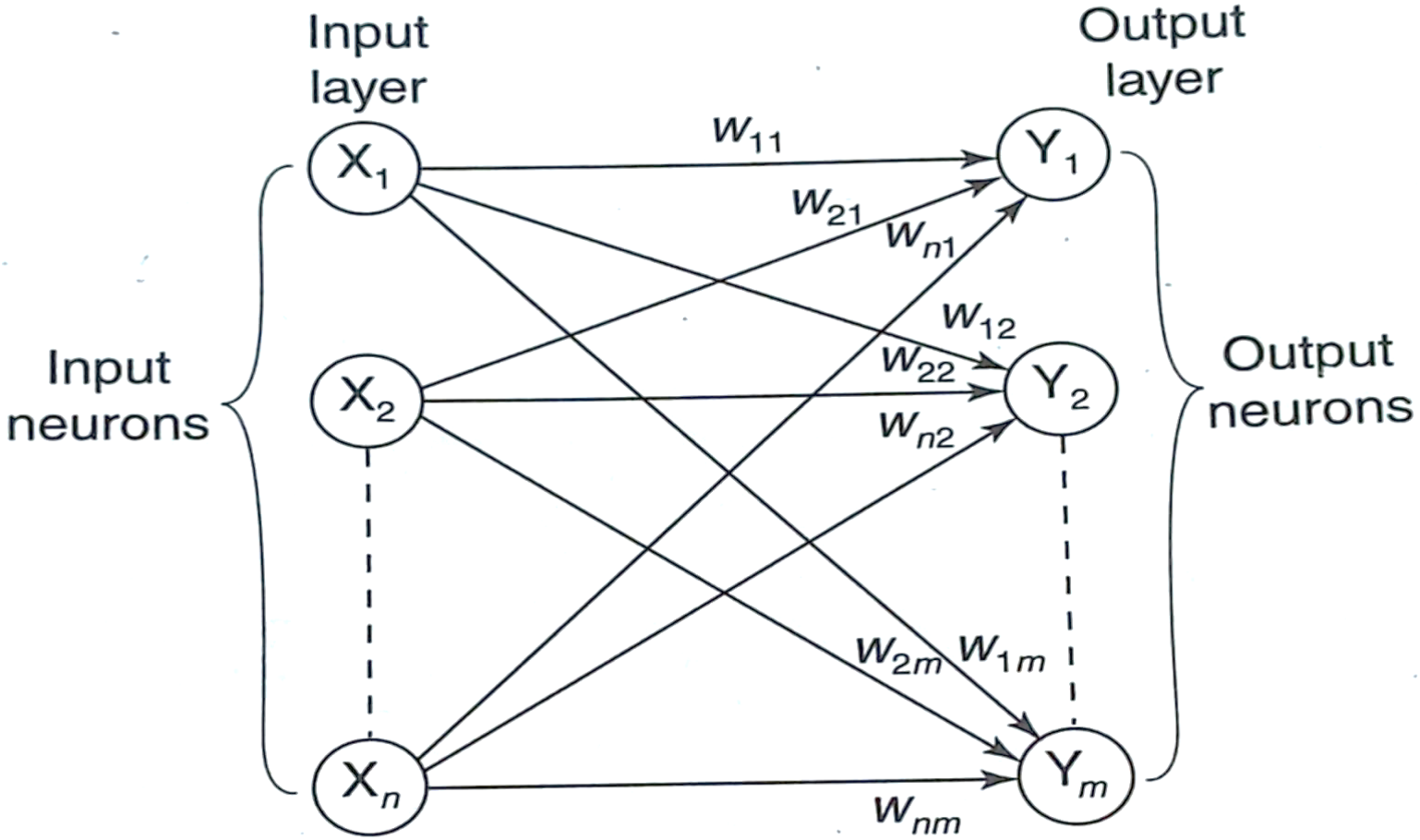
$w1=2.0, w2=-0.8, w3=0.6$

Apply **Tanh activation function**:

Architectures of ANN

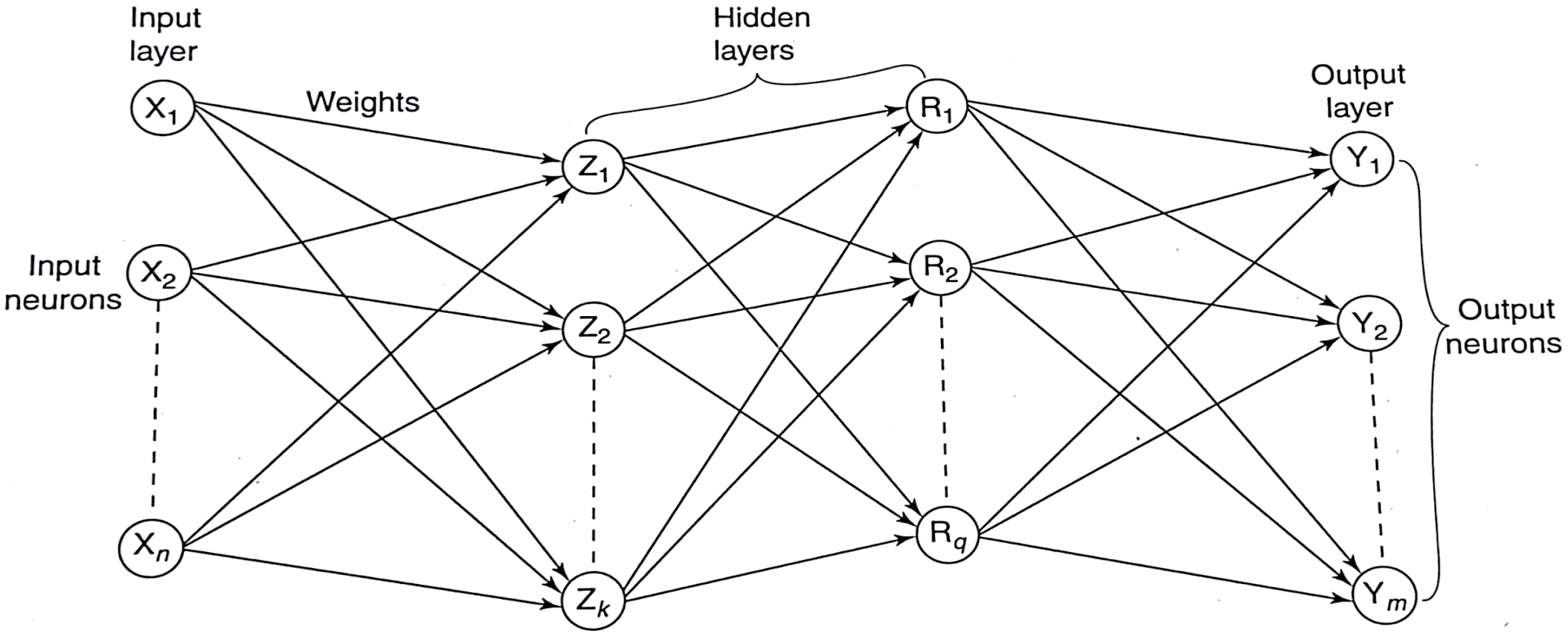
- Single-layer Feedforward Network
- Multi-layer Feedforward Network
- Convolutional Neural Network (CNN)
- Recurrent Neural Network (RNN)

Single-layer Feedforward Network



1. **Input to Output Connection:** The network has an input layer consisting of input neurons (denoted $X1, X2, ..., Xn$X 1 ,X 2 ,...,X n) that send signals directly to the output layer neurons ($Y1, Y2, ..., Ym$Y 1 ,Y 2 ,...,Y m). The data flows in one direction, from the input layer to the output layer, without any loops or cycles (hence "feedforward").
2. **Weighted Connections:** Each input neuron is connected to each output neuron via weighted connections (wijw ij). These weights determine the strength of the connection between the input and output neurons. The purpose of these weights is to modulate the influence of each input on the output.
3. **Linear Combination and Activation:** The input signals are combined in the output layer as a weighted sum, and each output neuron applies an activation function to this sum to produce the final output.
4. **No Hidden Layer:** This network is a single-layer model, meaning there is no hidden layer between the input and output. The network is simpler than more complex architectures like multilayer perceptron (MLPs) but still can solve tasks that are linearly separable.

Multi-layer Feedforward Network



Contd.

1. **Input Layer:** The network has input neurons (denoted $X1, X2, ..., Xn$X 1 ,X 2 ,...,X n) that receive data from the outside world. These neurons represent the features of the input data and feed into the network to be processed.
2. **Hidden Layers:** The network contains one or more hidden layers (denoted $Z1, Z2, ..., Zk$Z 1 ,Z 2 ,...,Z k).Hidden layers receive the weighted input from the input layer and pass it through activation functions to produce outputs for the next layer.The number of hidden layers and neurons can vary, which helps the network learn complex patterns and non-linear relationships.
3. **Weights:** The connections between layers are weighted (shown as weights between XX and ZZ layers, and between ZZ and YY layers).Weights determine the strength of the connections and are adjusted during training using optimization techniques like backpropagation.
4. **Output Layer:** The final output layer (denoted $Y1, Y2, ..., Ym$Y 1 ,Y 2 ,...,Y m) produces the result of the network's processing, which could be for tasks like classification or regression. The output is calculated by processing the results of the hidden layers through another set of weights and activation functions.

Contd.

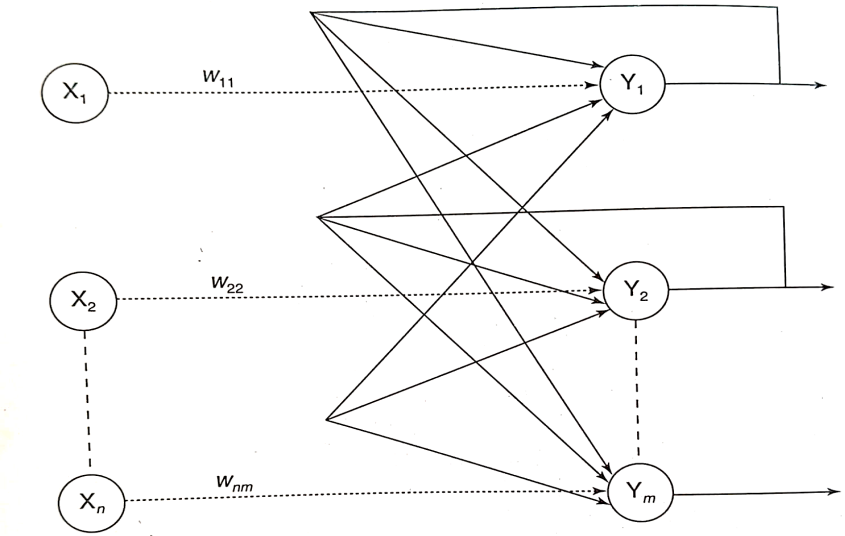
5. **Feedforward Process:** The information in this network flows forward from the input layer to the output layer through the hidden layers. There are no feedback loops or cycles, making it a feedforward network.
6. **Activation Functions:** Activation functions (like Sigmoid, ReLU, or Tanh) are applied to the weighted sum of inputs at each layer to introduce non-linearity, allowing the network to learn complex patterns.
7. **Training and Learning:** The network is trained using backpropagation to adjust weights based on the error between the predicted and actual outputs, optimizing the network's performance.

Convolutional Neural Network (CNN)

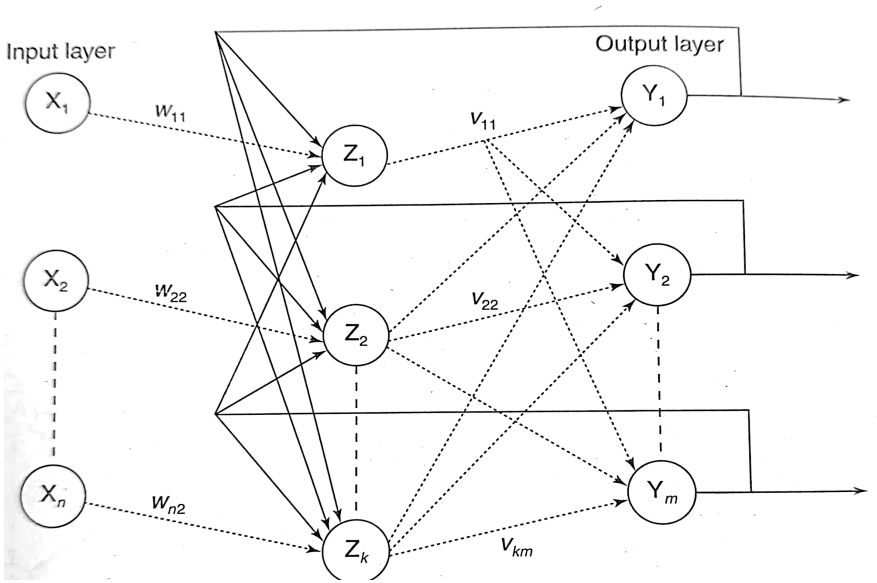
- CNNs are similar to feedforward networks.
- But they’re usually utilized for image recognition, pattern recognition, and/or computer vision.
- It helps computers recognize patterns in images, like identifying objects or faces.
- The model is designed to automatically learn spatial hierarchies of features from images or other grid-like data for tasks like classification and object detection.

Recurrent Neural Network (RNN)

- They are identified by their feedback loops.
- These learning algorithms are primarily leveraged when using time-series data to predict future outcomes, such as stock market predictions or sales forecasting.



Single-Layer Recurrent Network



Multi-Layer Recurrent Network

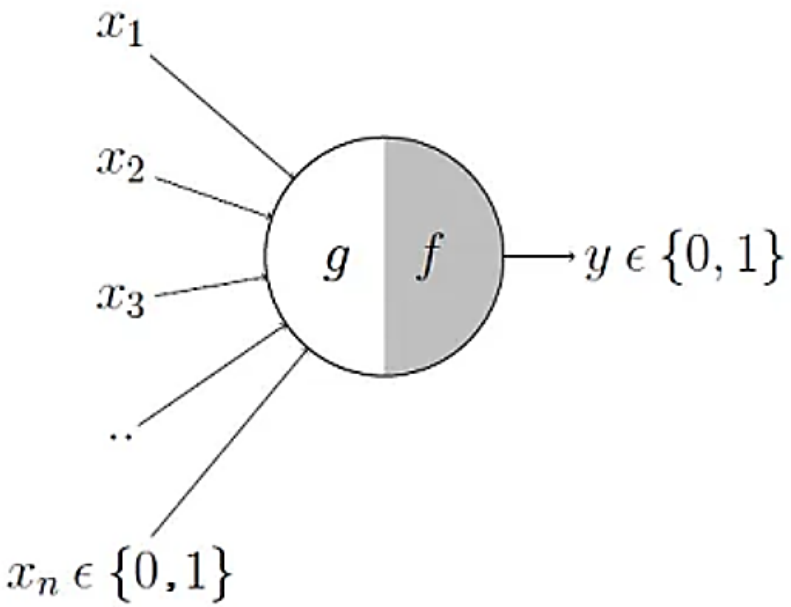
Advantages of ANN

- ANNs exhibits mapping capabilities, that is, they can map input patterns to their associated output pattern.
- The ANNs learn by examples. Thus, an ANN architecture can be trained with known example of a problem before they are tested for their inference capabilities on unknown instance of the problem. In other words, they can identify new objects previous untrained.
- The ANNs posses the capability to generalize. This is the power to apply in application where exact mathematical model to problem are not possible.

- The ANNs are robust system and fault tolerant. They can therefore, recall full patterns from incomplete, partial or noisy patterns.
- The ANNS can process information in parallel, at high speed and in a distributed manner. Thus a massively parallel distributed processing system made up of highly interconnected (artificial) neural computing elements having ability to learn and acquire knowledge is possible.

McCulloch-Pitts Neuron Model

- Mankind's first mathematical model of a biological neuron.
- The first computational model of a neuron was proposed by Warren McCulloch (neuroscientist) and Walter Pitts (logician) in 1943.



The first part, **g** takes an input, performs an aggregation and based on the aggregated value the second part, **f** makes a decision.

- Let's suppose that I want to predict my own decision, **whether to watch a random football game or not on TV**.
- The inputs are all boolean i.e., {0,1} and my output variable is also boolean **{0: Will watch it, 1: Won't watch it}.**
- So, ***x_1*** could be *isPremierLeagueOn* (I like Premier League more)
- ***x_2*** could be *isItAFriendlyGame* (I tend to care less about the friendlies)
- ***x_3*** could be *isNotHome* (Can't watch it when I'm running errands. Can I?)
- ***x_4*** could be *isManUnitedPlaying* (I am a big Man United fan. GGMU!) and so on.

- **These inputs can either be *excitatory* or *inhibitory*.**

- **Inhibitory inputs are those that have maximum effect on the decision making irrespective of other inputs** i.e., if ***x_3*** is 1 (not home) then my output will always be 0 i.e., the neuron will never fire, so ***x_3*** is an inhibitory input.
- **Excitatory inputs are NOT the ones that will make the neuron fire on their own but they might fire it when combined together**.
- Formally, this is what is going on:

$$g(x_1, x_2, x_3, ..., x_n) = g(\mathbf{x}) = \sum_{i=1}^{n} x_i$$

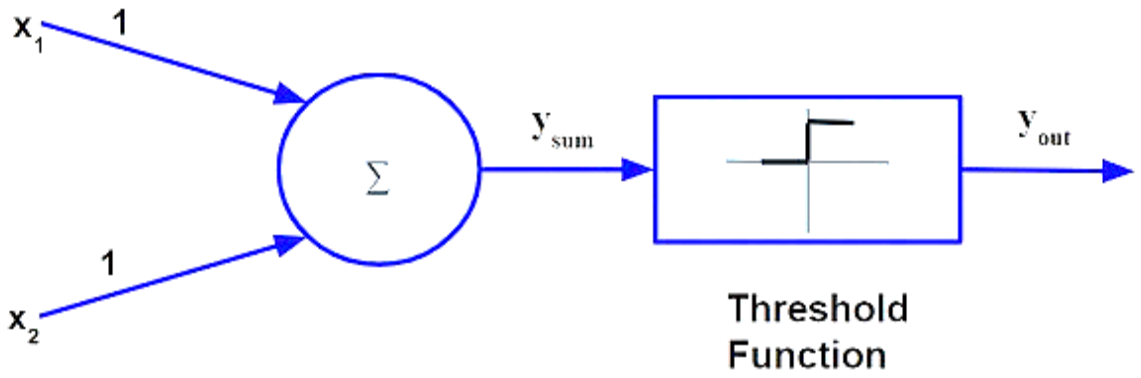
$$y = f(g(\mathbf{x})) = 1 \quad if \quad g(\mathbf{x}) \geq \theta$$
$$= 0 \quad if \quad g(\mathbf{x}) < \theta$$

- **g(x)** is just doing a sum of the inputs — a simple aggregation.
- And theta here is called **thresholding parameter**.
- For example, if I always watch the game when the sum turns out to be 2 or more, the theta is 2 here. This is called the Thresholding Logic.

- **Example:**
  - John carries an umbrella if it is sunny or if it is raining. There are four given situations. I need to decide when John 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

- 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:

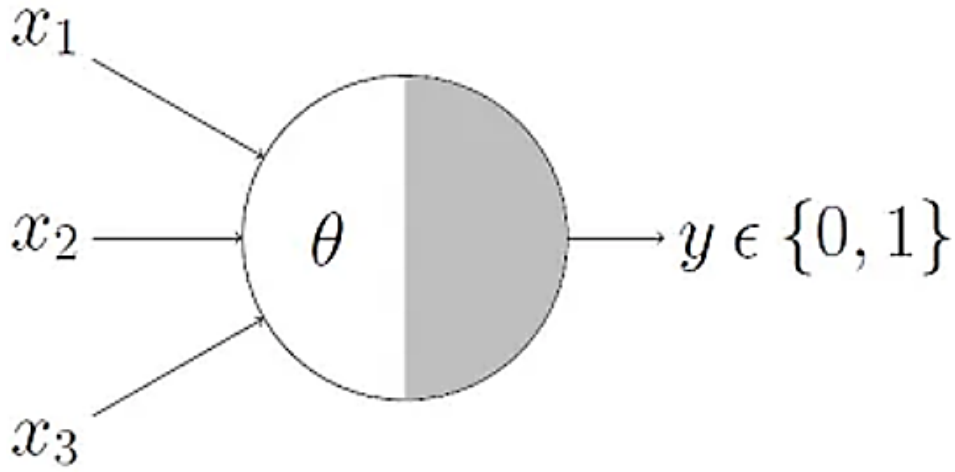


- Truth Table for this case will be:

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

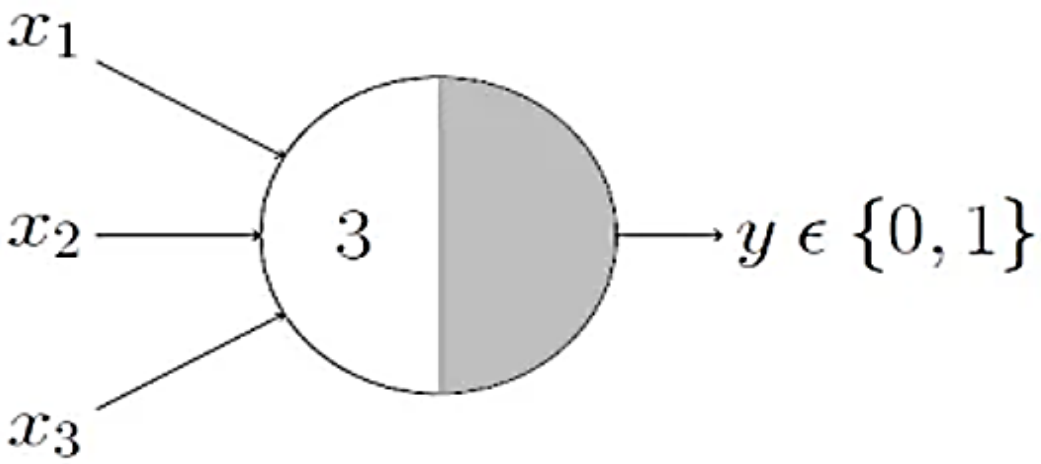
Boolean Functions Using M-P Neuron

- Till now the concise representation is like:



- This representation just denotes that, for the boolean inputs ***x_1***, ***x_2*** and ***x_3*** if the ***g*(x)** i.e., **sum ≥ theta**, the neuron will fire otherwise, it won't.

AND Function



An AND function neuron would only fire when ALL the inputs are ON i.e., $g(\mathbf{x}) \geq 3$ here.

TABLE 1

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

In McCulloch–Pitts neuron, only analysis is being performed. Hence, assume the weights be $w_1 = 1$ and $w_2 = 1$. The network architecture is shown in Figure 4. With these assumed weights, the net input is calculated for four inputs: For inputs

$$(1, 1),\ y_{in} = x_1 w_1 + x_2 w_2 = 1 \times 1 + 1 \times 1 = 2$$

$$(1, 0),\ y_{in} = x_1 w_1 + x_2 w_2 = 1 \times 1 + 0 \times 1 = 1$$

$$(0, 1),\ y_{in} = x_1 w_1 + x_2 w_2 = 0 \times 1 + 1 \times 1 = 1$$

$$(0, 0),\ y_{in} = x_1 w_1 + x_2 w_2 = 0 \times 1 + 0 \times 1 = 0$$

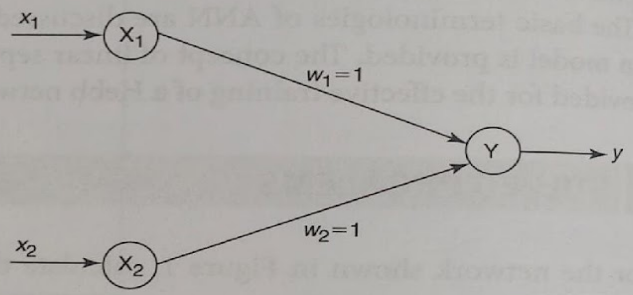


Figure 4 Neural net.

For an AND function, the output is high if both the inputs are high. For this condition, the net input is calculated as 2. Hence, based on this net input, the threshold is set, i.e. if the threshold value is greater than or equal to 2 then the neuron fires, else it does not fire. So the threshold value is set equal to $2(\theta = 2)$. This can also be obtained by

$$\theta \geq nw - p$$

Here, $n = 2$, $w = 1$ (excitatory weights) and $p = 0$ (no inhibitory weights). Substituting these values in the above-mentioned equation we get

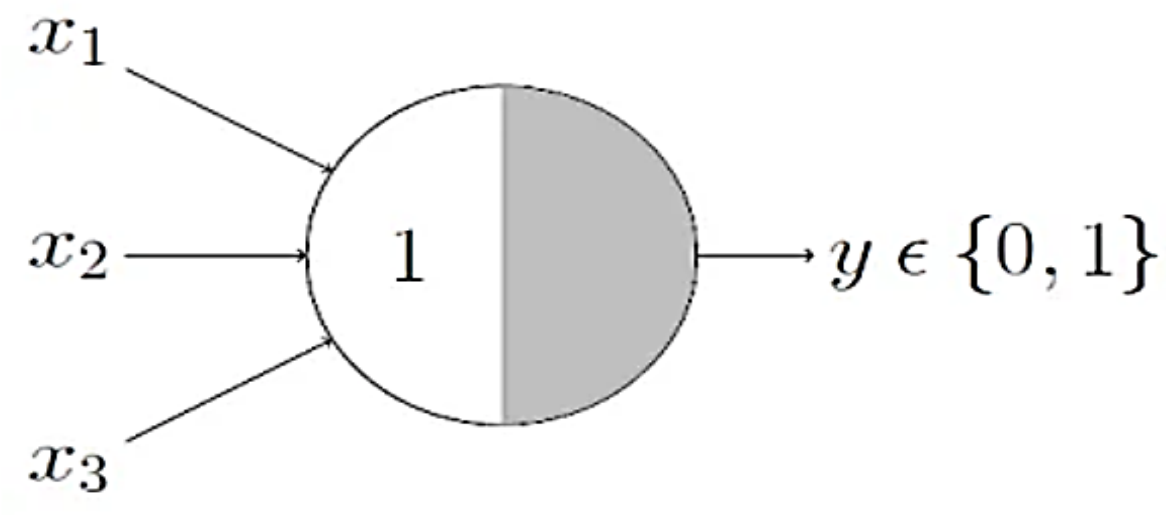
$$\theta \geq 2 \times 1 - 0 \Rightarrow \theta \geq 2$$

Thus, the output of neuron Y can be written as

$$y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} \geq 2 \\ 0 & \text{if } y_{in} < 2 \end{cases}$$

where "2" represents the threshold value.

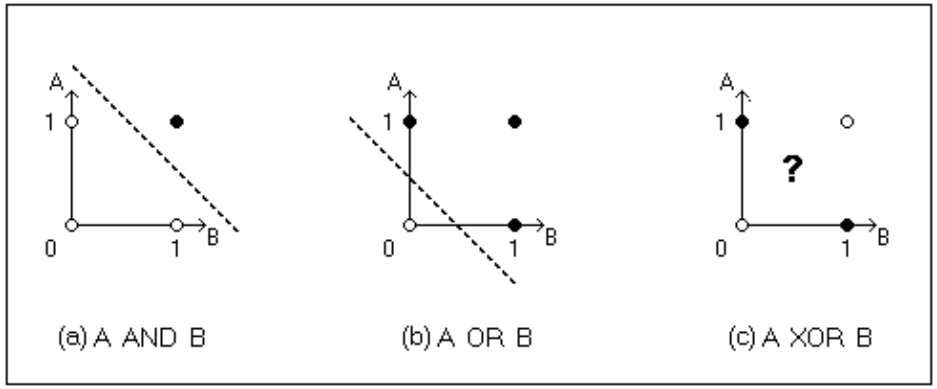
OR Function



I believe this is self explanatory as we know that an OR function neuron would fire if ANY of the inputs is ON i.e., $g(\mathbf{x}) \geq 1$ here.

Limitations Of M-P Neuron

- What about non-boolean (say, real) inputs?
- Do we always need to hand code the threshold?
- Are all inputs equal? What if we want to assign more importance to some inputs?
- What about functions which are not linearly separable? Say XOR function.



Hebb Network uses Bipolar Data

- The Hebb rule is more suited for the bipolar date (-1,1) than binary data(0,1)
- If binary data is used, the above weight updation formula cannot distinguish two conditions namely:
  1. A training pair in which an input unit is “on” and target value is “off”.
  2. A training pair in which both the input unit and target value are “off”.

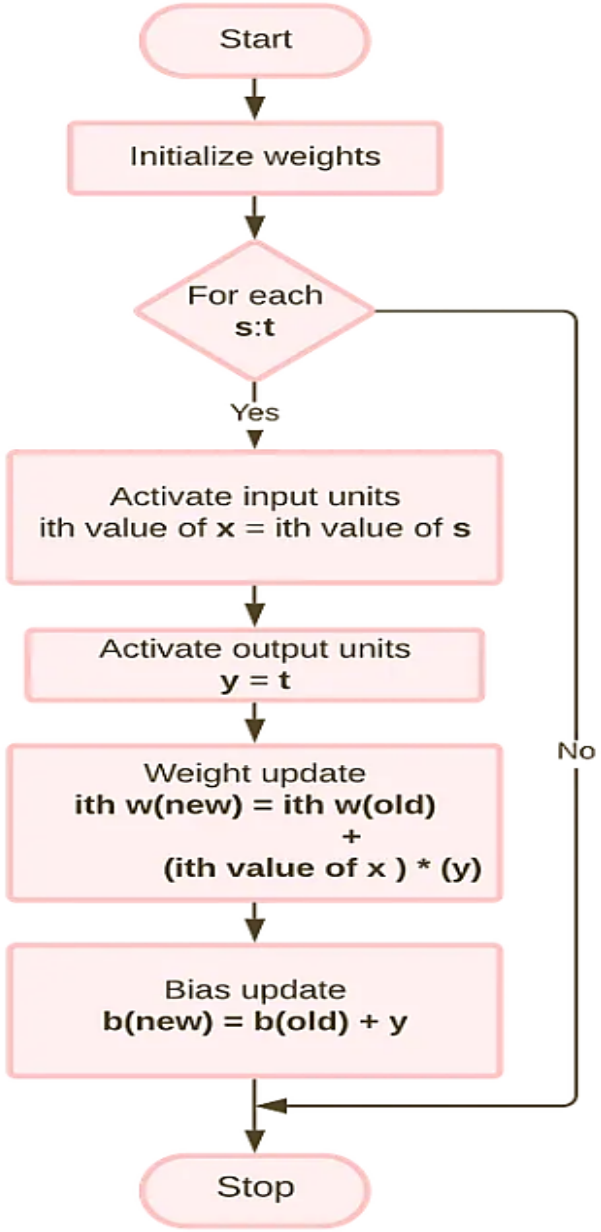
Intuition Behind Hebbian Rule

The idea is:

- ***"If input and output are correlated (both active or both inactive), reinforce the connection."***
- ***"If they are anti-correlated (one is active, the other not), weaken the connection."***

X_i	Y_i	$X_i \cdot Y_i$	**Interpretation**	**Δw**
+1	+1	+1	Both active	↑
-1	-1	+1	Both inactive	↑
+1	-1	-1	Input on, target off	↓
-1	+1	-1	Input off, target on	↓

Flowchart of Training Algorithm



1. Initialize the weights and bias to ‘0’ i.e w1=0,w2=0, …., wn=0.
2. Steps 2–4 have to be performed for each input training vector and target output pair i.e. s:t (s=training input vector, t=training output vector)
3. Input units' activation are set. Generally, the activation function of input layer is identity function $\mathbf{x_i = s_i}$ for i=1 to n
4. Output units' activations are set: **y=t**
5. Weight adjustments and bias adjustments are performed;
   - $w_i(new) = w_i(old) + x_i y$
   - b(new) = b(old) + y

8. Design a Hebb net to implement logical AND function (use bipolar inputs and targets).

Solution: The training data for the AND function is given in Table 9.

TABLE 9

Inputs			Target
x_1	x_2	b	y
1	1	1	1
1	−1	1	−1
−1	1	1	−1
−1	−1	1	−1

The network is trained using the Hebb network training algorithm discussed in Section 2.7.3. Initially the weights and bias are set to zero, i.e.,

$$w_1 = w_2 = b = 0$$

- **First input $[x_1\ x_2\ b] = [111]$ and target = 1 [i.e., $y = 1$]:** Setting the initial weights as old weights and applying the Hebb rule, we get

$$w_i(\text{new}) = w_i(\text{old}) + x_i y$$

$$w_1(\text{new}) = w_1(\text{old}) + x_1 y = 0 + 1 \times 1 = 1$$

$$w_2(\text{new}) = w_2(\text{old}) + x_2 y = 0 + 1 \times 1 = 1$$

$$b(\text{new}) = b(\text{old}) + y = 0 + 1 = 1$$

The weights calculated above are the final weights that are obtained after presenting the first input. These weights are used as the initial weights when the second input pattern is presented. The weight change here is $\Delta w_i = x_i y$. Hence weight changes relating to the first input are

$$\Delta w_1 = x_1 y = 1 \times 1 = 1$$

$$\Delta w_2 = x_2 y = 1 \times 1 = 1$$

$$\Delta b = y = 1$$

- **Second input $[x_1\ x_2\ b] = [1\ -1\ 1]$ and $y = -1$:** The initial or old weights here are the final (new) weights obtained by presenting the first input pattern, i.e.,

The weight change here is

$$\Delta w_1 = x_1 y = 1 \times -1 = -1$$
$$\Delta w_2 = x_2 y = -1 \times -1 = 1$$
$$\Delta b = y = -1$$

The new weights here are

$$w_1(\text{new}) = w_1(\text{old}) + \Delta w_1 = 1 - 1 = 0$$
$$w_2(\text{new}) = w_2(\text{old}) + \Delta w_2 = 1 + 1 = 2$$
$$b(\text{new}) = b(\text{old}) + \Delta b = 1 - 1 = 0$$

Similarly, by presenting the third and fourth input patterns, the new weights can be calculated. Table 10 shows the values of weights for all inputs.

TABLE 10

Inputs			Weight changes				Weights		
x_1	x_2	b	y	Δw_1	Δw_2	Δb	w_1 (0	w_1 0	b 0)
1	1	1	1	1	1	1	1	1	1
1	−1	1	−1	−1	1	−1	0	2	0
−1	1	1	−1	1	−1	−1	1	1	−1
−1	−1	1	−1	1	1	−1	2	2	−2

The network can be represented as shown in Figure 12.

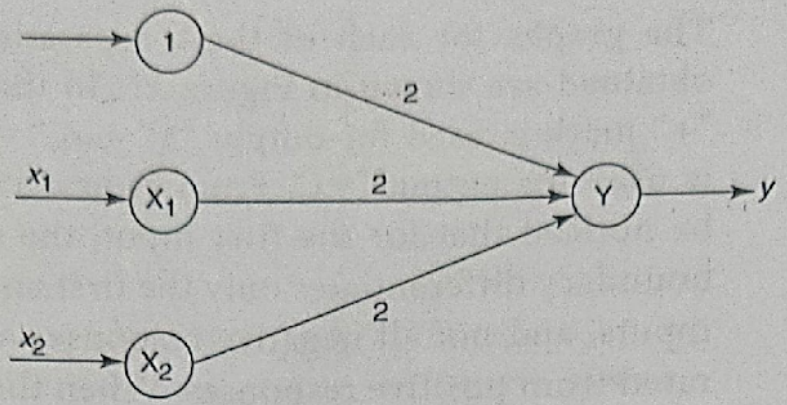


Figure 12 Hebb net for AND function.

9. Design a Hebb net to implement OR function (consider bipolar inputs and targets).

Solution: The training pair for the OR function is given in Table 11.

TABLE 11

Inputs			Target
x_1	x_2	b	y
1	1	1	1
1	−1	1	1
−1	1	1	1
−1	−1	1	−1

Initially the weights and bias are set to zero, i.e.,

$$w_1 = w_2 = b = 0$$

The network is trained and the final weights are outlined using the Hebb training algorithm discussed in Section 2.7.3. The weights are considered as final weights if the boundary line obtained from these weights separates the positive response region and negative response region.

By presenting all the input patterns, the weights are calculated. Table 12 shows the weights calculated for all the inputs.

TABLE 12

Inputs				Weight changes			Weights		
x_1	x_2	b	y	Δw_1	Δw_2	Δb	w_1 (0	w_2 0	b 0)
1	1	1	1	1	1	1	1	1	1
1	−1	1	1	1	−1	1	2	0	2
−1	1	1	1	−1	1	1	1	1	3
−1	−1	1	−1	1	1	−1	2	2	2

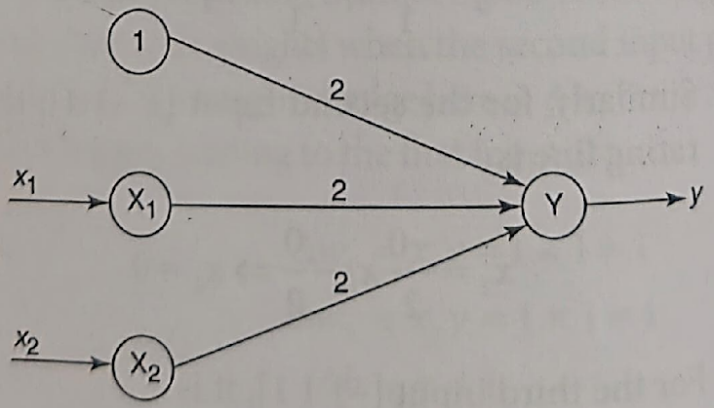


Figure 14 Hebb net for OR function.

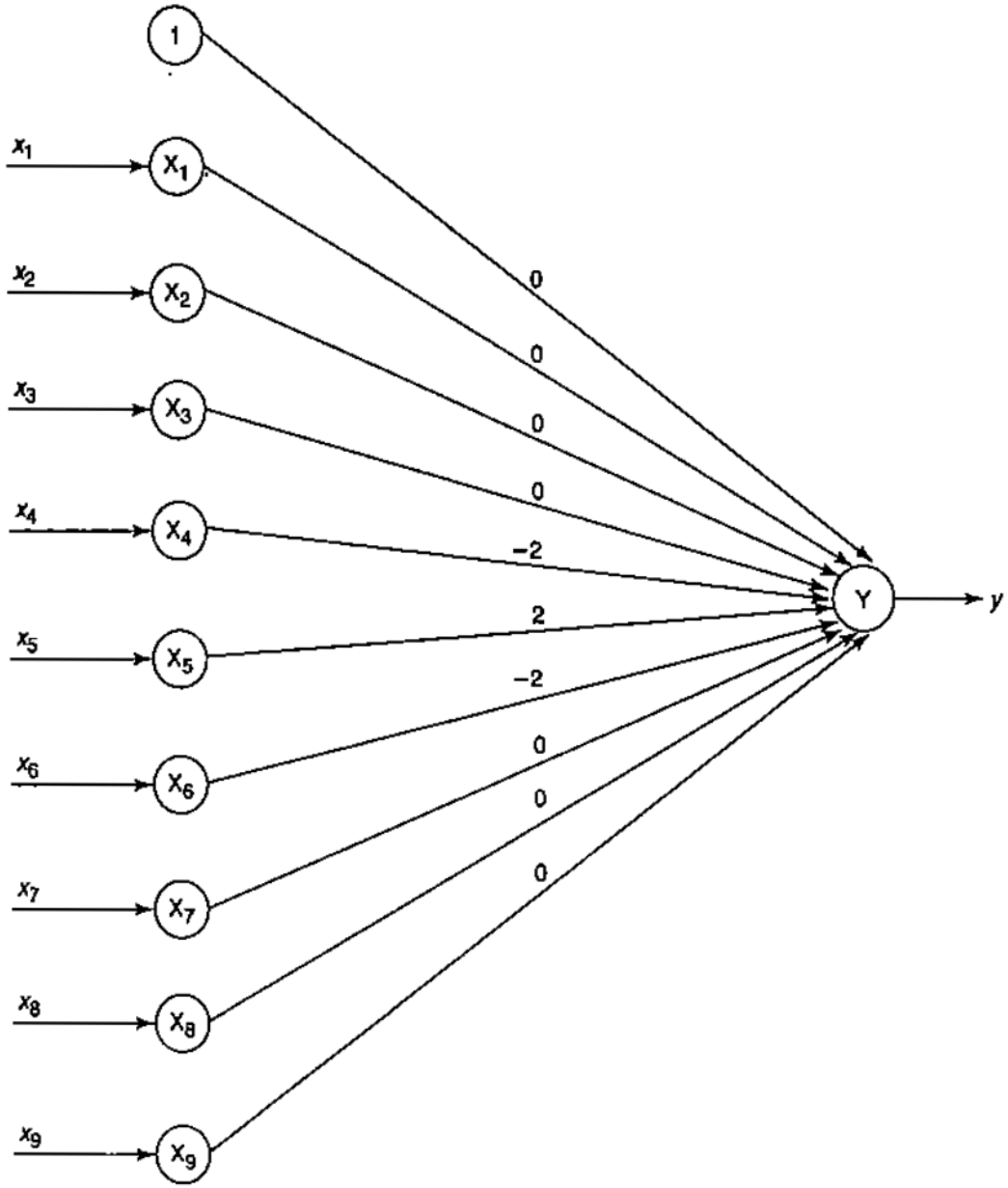
Draw hebb network.

1.

Pattern	Inputs										Target
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	b	y
I	1	1	1	−1	1	−1	1	1	1	1	1
O	1	1	1	1	−1	1	1	1	1	1	−1

2.

Pattern	Inputs										Target
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	b	y
L	1	−1	−1	1	−1	−1	1	1	1	1	1
U	1	−1	1	1	−1	1	1	1	1	1	−1


1
x_1
X_1
x_2
X_2
x_3
X_3
x_4
X_4
x_5
X_5
x_6
X_6
x_7
X_7
x_8
X_8
x_9
X_9
0
0
0
0
−2
2
−2
0
0
0
Y
y

Perceptron: The Simplest Neural Unit

- A **perceptron** is the earliest form of a neural network unit, introduced by **Frank Rosenblatt** in 1958.
- It is a **binary classifier** that makes predictions based on a **linear combination** of input features.
- The perceptron algorithm was one of the first algorithms used to implement a simple neural network.

Components of a Perceptron

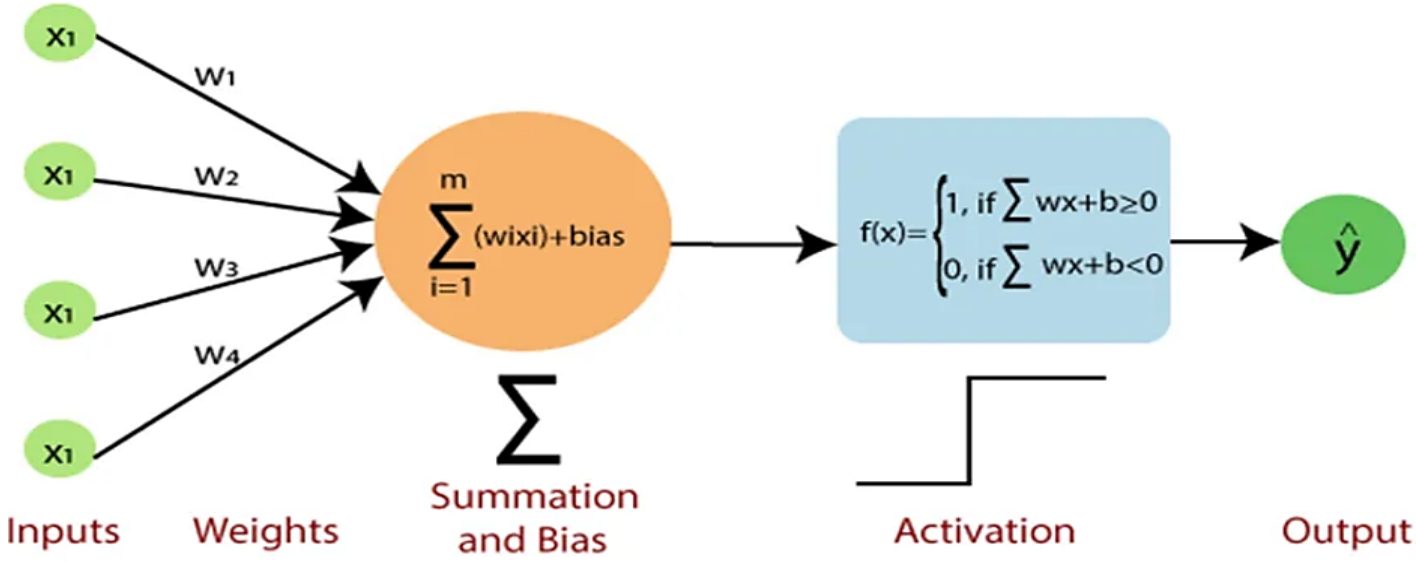
- **Inputs:** The perceptron takes several inputs (x1,x2,…,xn)
- **Weights:** Each input is associated with a weight (w1,w2,…,wn).
- **Bias:** A bias term (b) is **added to shift the decision boundary**.
- **Activation Function:** The perceptron uses a **step function** (a simple thresholding function) to determine whether the weighted sum of inputs plus the bias is above or below a certain threshold.
- The mathematical representation is:

$$y = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} w_i x_i + b > 0 \\ 0 & \text{if } \sum_{i=1}^{n} w_i x_i + b \leq 0 \end{cases}$$

- **Binary Output:** The output of a perceptron is binary (1 or 0), making it suitable for linearly separable classification problems.

Limitations of Perceptron

- **Linear separability:** The perceptron can only solve problems that are linearly separable (i.e., it can only classify data that can be separated by a straight line or hyperplane). **It cannot solve more complex problems like XOR.**
- **Single-layer model:** The original perceptron is a single-layer model and does not have hidden layers, limiting its expressiveness.



Neuron: A Generalized Unit in Neural Networks

- A **neuron**, or artificial neuron, is a more generalized version of the perceptron and is the building block of modern deep learning architectures.
- Neurons in deep learning are part of **multi-layer neural networks**, which can have multiple hidden layers.

Key Differences and Features of a Neuron

- **Activation Function:** Unlike the perceptron, which uses a simple step function for activation, *neurons in modern neural networks can use a variety of activation functions, such as sigmoid, tanh, ReLU.*

- **Multi-layer Networks:** Neurons are part of more sophisticated architectures called ***multi-layer perceptrons (MLPs)*** *or deep neural networks, where neurons are organized into layers (input layer, hidden layers, and output* layer). Each layer performs computations, and the output of one layer is fed as input to the next.

Contd.

- **Continuous Output:** Neurons can *output continuous values*, unlike the binary output of a perceptron. This makes them more versatile for tasks like *regression, multi-class classification, and complex feature extraction.*

- **Learning through Backpropagation:** Neurons in deep learning models are trained using backpropagation, which *adjusts the weights based on the error between the predicted and actual outputs.* Whereas the perceptron uses a simpler update rule that works only for linearly separable problems.

Key Differences Between Perceptron and Neuron in Deep Learning

Aspect	Perceptron	Neuron in Deep Learning
Output	Binary (0 or 1)	Continuous or non-linear (depends on the activation function)
Activation Function	Step function	Sigmoid, ReLU, Tanh, Softmax, etc.
Complexity	Simple, handles linearly separable problems	Complex, handles non-linear, multi-dimensional problems
Layers	Single layer (original perceptron)	Multi-layer (input, hidden, and output layers)
Learning	Basic learning rule for linear problems	Gradient descent with backpropagation
Use Case	Binary classification	Classification, regression, and various deep learning tasks

Difference between SLP and MLP

Aspect	Single-Layer Perceptron (SLP)	Multi-Layer Perceptron (MLP)
Architecture	One input layer, one output layer (no hidden layers)	Input, hidden, and output layers
Problem Solvability	Solves only linearly separable problems (e.g., AND)	Solves both linear and non-linear problems (e.g., XOR)
Activation Function	Step function (binary output)	Non-linear functions (e.g., sigmoid, ReLU, tanh)
Learning Algorithm	Perceptron learning rule (no backpropagation)	Trained using backpropagation and gradient descent
Output	Binary (0 or 1)	Continuous (for regression) or multi-class (for classification)
Applications	Simple binary classification	Complex tasks (classification, regression, image recognition)
Complexity	Simple and limited	Complex and powerful

Dataset

- A dataset is a **collection of related data** organized in a structured way, usually in the form of rows and columns, that is used for analysis, training machine learning models, or drawing conclusions.

Student ID	Name	Grade	Attendance (%)
101	Ayesha	85	92
102	Ravi	78	88
103	Fatima	91	95

- This table is a **dataset** where:
  - ? Each **row** = a data entry or record (101, Ayesha, 85, 92)
  - ? Each **column** = a feature (like Student ID, Name, Grade, Attendance (%))

Labelled Dataset

- A **labeled dataset** includes **input data** *and* the **correct output (label or target value)** for each example.

Square Feet	Bedrooms	Bathrooms	Location	Age (Years)	Price ($)
1400	3	2	New York	10	550,000
1600	4	2	Chicago	5	460,000
1200	2	1	Dallas	15	300,000
1800	3	2	San Diego	7	620,000
2000	4	3	Seattle	2	750,000

- The first **5 columns** are the **features (inputs)**.
- The last column **(Price)** is the label **(output)** — this is what the model learns to predict.

Unlabelled Dataset

- An **unlabeled dataset** includes only **input data**, with **no labels** or correct answers provided.

Square Feet	Bedrooms	Bathrooms	Location	Age (Years)
1400	3	2	New York	10
1600	4	2	Chicago	5
1200	2	1	Dallas	15
1800	3	2	San Diego	7
2000	4	3	Seattle	2

- This dataset has **no labels (i.e., no price column)**.
- It includes **only input features.**

Independent Variable

- Also called as **input, feature, or predictors**.
- The variable that you **control** or use to **predict** something else.
- These are the columns that **affect or influence** the outcome.

*"What you **change or know** to try to predict something."*

- In house price prediction example, the **Independent Variables** are Square Feet, Bedrooms, Location, Age.

Dependent Variable

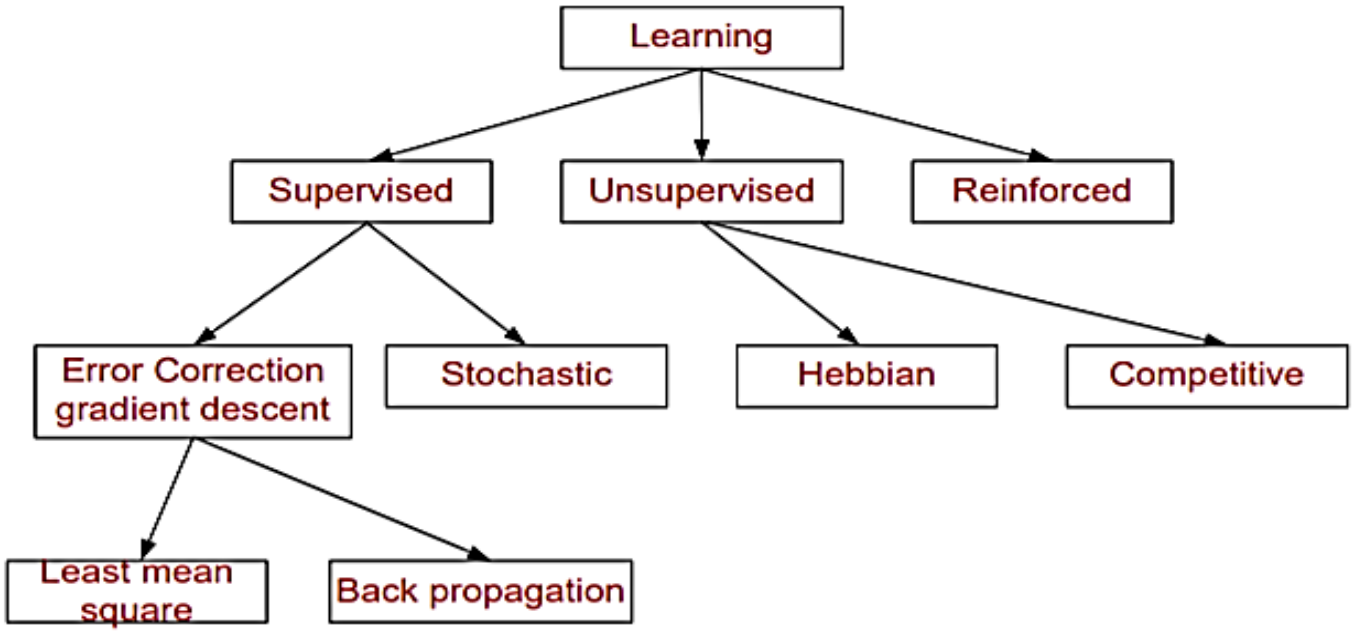
- Also called **label**, **target**, or **output**.
- The variable you are trying to **predict or explain**.
- It **depends on** the values of the independent variables.

*"What you want to **find out or predict**."*

- In house price prediction example, **Dependent Variable is Price.**
- Because the price **depends on** the size, number of bedrooms, location, and age.

Types Of Learning

- There are several learning techniques.
- A taxonomy of well known learning techniques are shown in the following.



Supervised learning

- Supervised learning is a type of machine learning method in which we provide sample **labelled data** to the machine learning system in order to train it, and on that basis, it predicts the output.
- The system creates a model using labelled data to understand the datasets and learn about each data
- Once the training and processing are done then we test the model by providing a sample data to check whether it is predicting the exact output or not.

Contd.

User ID	Gender	Age	Salary	Purchased
15624510	Male	19	19000	0
15810944	Male	35	20000	1
15668575	Female	26	43000	0
15603246	Female	27	57000	0
15804002	Male	19	76000	1
15728773	Male	27	58000	1
15598044	Female	27	84000	0
15694829	Female	32	150000	1
15600575	Male	25	33000	1
15727311	Female	35	65000	0
15570769	Female	26	80000	1
15606274	Female	26	52000	0
15746139	Male	20	86000	1
15704987	Male	32	18000	0
15628972	Male	18	82000	0
15697686	Male	29	80000	0
15733883	Male	47	25000	1

Figure A: CLASSIFICATION

Temperature	Pressure	Relative Humidity	Wind Direction	Wind Speed
10.69261758	986.882019	54.19337313	195.7150879	3.278597116
13.59184184	987.8729248	48.0648859	189.2951202	2.909167767
17.70494885	988.1119385	39.11965597	192.9273834	2.973036289
20.95430404	987.8500366	30.66273218	202.0752869	2.965289593
22.9278274	987.2833862	26.06723423	210.6589203	2.798230886
24.04233986	986.2907104	23.46918024	221.1188507	2.627005816
24.41475295	985.2338867	22.25082295	233.7911987	2.448749781
23.93361956	984.8914795	22.35178837	244.3504333	2.454271793
22.68800023	984.8461304	23.7538641	253.0864716	2.418341875
20.56425726	984.8380737	27.07867944	264.5071106	2.318677425
17.76400389	985.4262085	33.54900114	280.7827454	2.343950987
11.25680746	988.9386597	53.74139903	68.15406036	1.650191426
14.37810685	989.6819458	40.70884681	72.62069702	1.553469896
18.45114201	990.2960205	30.85038484	71.70604706	1.005017161
22.54895853	989.9562988	22.81738811	44.66042709	0.264133632
24.23155922	988.796875	19.74790765	318.3214111	0.329656571

Figure B: REGRESSION

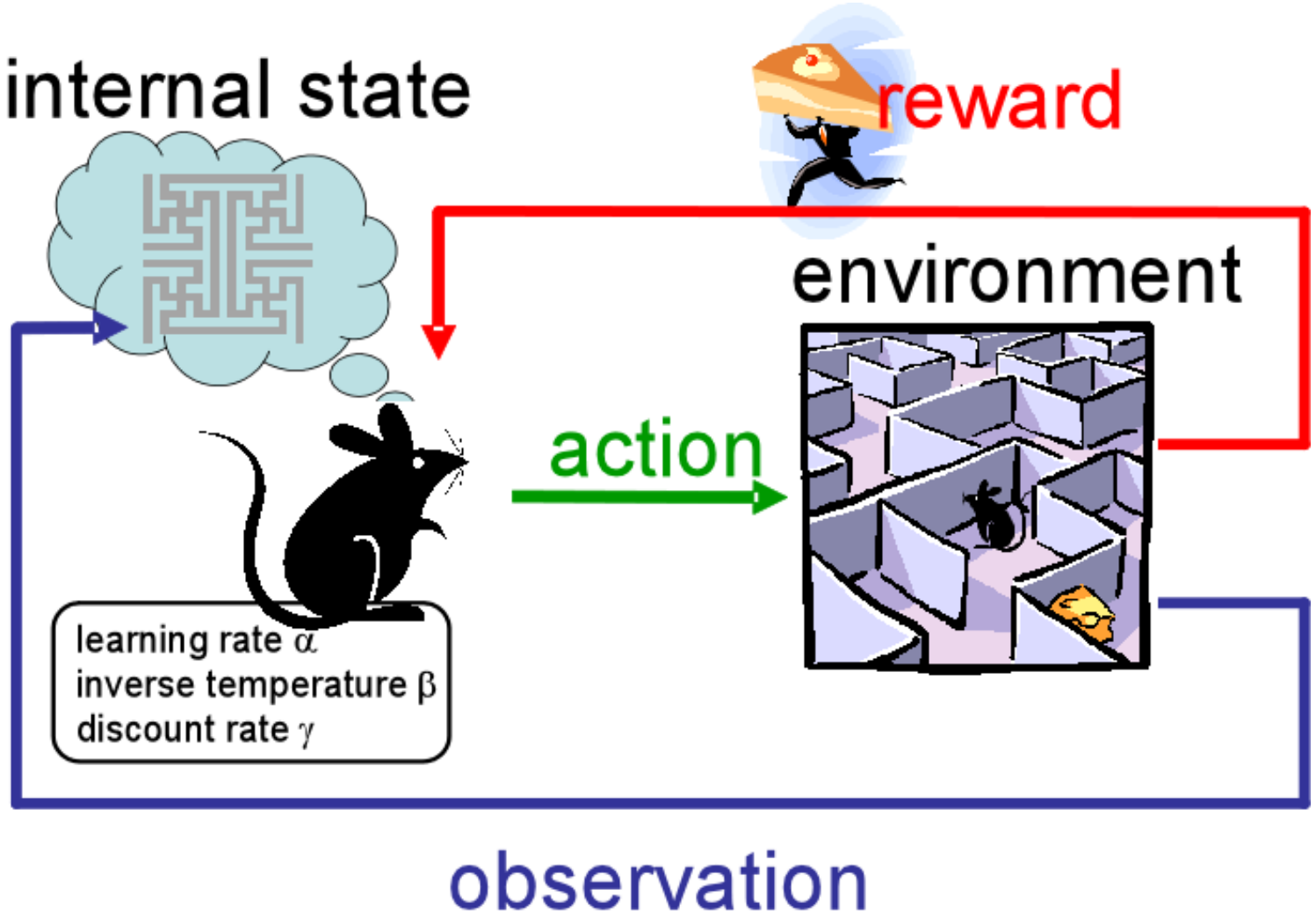
Unsupervised learning

- Unsupervised learning is a learning method in which a machine learns without any supervision.
- The training is provided to the machine with the **set of data that has not been labelled**, classified, or categorized, and the algorithm needs to act on that data without any supervision.
- The goal of unsupervised learning is to restructure the input data into new features or a group of objects with similar patterns.
- In unsupervised learning, **we don't have a predetermined result**.
- The machine tries to find useful insights from the huge amount of data.
- It can be further classifieds into two categories of algorithms:
  - **Clustering**
  - **Association**

Reinforcement Learning

- Reinforcement learning is a **feedback-based learning method**, in which a learning agent gets a reward for each right action and gets a penalty for each wrong action.
- The agent learns automatically with these feedbacks and improves its performance.
- In reinforcement learning, **the agent interacts with the environment and explores it**.
- The goal of an agent is to get the most reward points, and hence, it improves its performance.
- The robotic dog, which automatically learns the movement of his arms, is an example of Reinforcement learning.

Contd.


internal state
reward
environment
action
learning rate α
inverse temperature β
discount rate γ
observation

Detect type of Learning

1.A university collects student data: attendance, assignment scores, exam marks. They want to build a system that can suggest whether a student is “likely to pass” or “likely to fail.”

2.A hospital installs a system that groups patients with similar symptoms together so doctors can detect unknown diseases. The system is never told the actual disease names.

Detect type of Learning

3.An app records how much time you spend on different social media posts (videos, memes, articles). It then learns to show you more of what keeps you engaged longer.

4.A company wants to forecast next month’s electricity demand using past records of daily consumption. The data includes numbers (in units), but not categories like “high” or “low.”

Detect type of Learning

5.A delivery drone tries different paths to reach the destination faster. When it chooses a short safe path, it gets a high score; when it takes too long or crashes, it loses points. Over time, it improves.

6.A bank feeds a model with old credit card transactions labeled as “fraud” or “not fraud.” The system should flag suspicious transactions in the future.

Detect type of Learning

7.A movie platform has thousands of films but no genre labels. It analyzes patterns in how people watch and groups films into hidden categories that “seem similar.”

8.A chess program tries millions of moves. At first, it plays randomly, but it learns which moves increase its chance of winning by playing repeatedly against itself.

Answers

1. Supervised – Classification
2. Unsupervised – Clustering
3. Reinforcement
4. Supervised – Regression
5. Reinforcement
6. Supervised – Classification
7. Unsupervised – Clustering
8. Reinforcement