



KTU STUDY MATERIALS | SYLLABUS | LIVE
NOTIFICATIONS | SOLVED QUESTION PAPERS

Website: www.ktunotes.in



Module 1:

Introduction to Soft Computing & Artificial Neural Network

- ▶ Introduction to Soft Computing.
- ▶ Difference between Hard Computing & Soft Computing.
- ▶ Applications of Soft Computing.
- ▶ Artificial Neurons Vs Biological Neurons.
- ▶ Basic models of artificial neural networks
 - ▶ Connections,
 - ▶ Learning,
 - ▶ Activation Functions.
- ▶ McCulloch and Pitts Neuron.
- ▶ Hebb network.

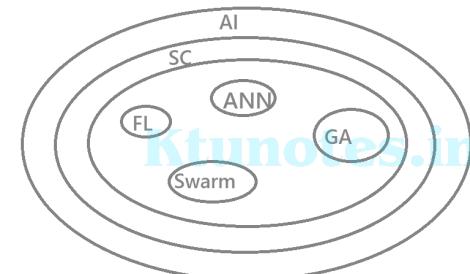
Ktunotes.in

Introduction to Soft Computing Ktunotes.in

Introduction to Soft Computing,

- ▶ Soft computing is an emerging collection of methodologies, which aim to exploit tolerance for **imprecision, uncertainty, and partial truth** to achieve robustness, tractability and total low cost.
- ▶ Role model of soft computing is **human mind**.
- ▶ Soft-computing is defined as a collection of techniques spanning many fields that fall under various categories in **computational intelligence**.
- ▶ Soft computing has three main branches:
 - ▶ fuzzy Systems,
 - ▶ artificial neural Networks
 - ▶ Genetic Algorithms

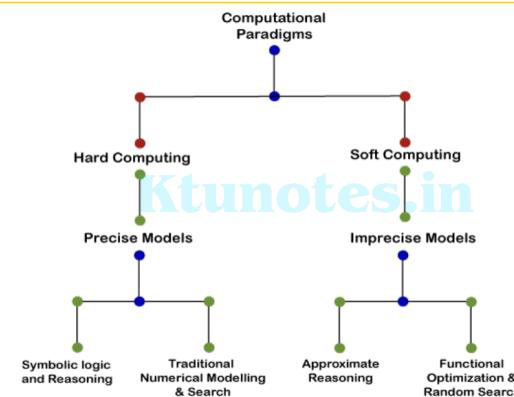
Evolution of Soft Computing



Problem Solving Methodologies

- ▶ Two major problem solving methodologies includes
 - ▶ **Hard computing**
 - ▶ Based on the concept of **precise modeling** and analyzing to yield **accurate results**.
 - ▶ Traditional AI uses Hard computing
 - ▶ Accurate Models based on Symbolic Logic Reasoning and numerical modelling
 - ▶ Works well for **simple problems**
 - ▶ **Soft computing**
 - ▶ Model the **Approximate System** and Uses inexact methods
 - ▶ Represents a computing which **reflects the human mind**.
 - ▶ Tolerant to imprecision, uncertainty, partial truth, and approximation.
 - ▶ **Well suited for real world problems** where ideal models are not available

Problem Solving Methodologies



soft computing

- ▶ The term soft computing was introduced by **Prof.Lotfi Zadeh**. (1994),University of California,Berkeley ,USA
- ▶ Goal
 - ▶ To emulate the human mind as closely as possible
 - ▶ Partnership with many fields
 - ▶ Neural network
 - ▶ Genetic Algorithm
 - ▶ Fuzzy Logic

Ktunotes.in

Hard computing

- ▶ In 1996, L. A. Zadeh (LAZ) introduced the term hard computing.
- ▶ According to LAZ: We term a computing as Hard computing,
- ▶ Characteristics of hard computing
 - ▶ Precise result is guaranteed.
 - ▶ Control action is unambiguous.
 - ▶ Control action is formally defined (i.e., with mathematical model or algorithm)..
- ▶ Examples 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 a graph, finding closet pair of points given a set of points, etc.).

Ktunotes.in

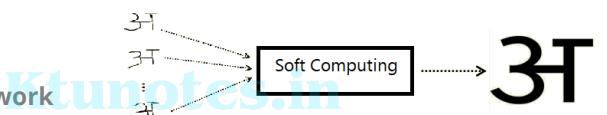
Soft computing

- ▶ The term soft computing was proposed by the inventor of fuzzy logic, Lotfi A. Zadeh
- ▶ Definition 1:
 - ▶ Soft computing 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, neurocomputing, and probabilistic reasoning.
- ▶ The role model for soft computing is the human mind
- ▶ Characteristics of soft computing
 - ▶ It does not require any mathematical modeling of problem solving.
 - ▶ It may not yield the precise solution.
 - ▶ Algorithms are adaptive (i.e., it can adjust to the change of dynamic environment).
 - ▶ Use some biological inspired methodologies such as genetics, evolution, Ant's behaviors, particles swarming, human nervous system, etc.).

Ktunotes.in

Soft computing

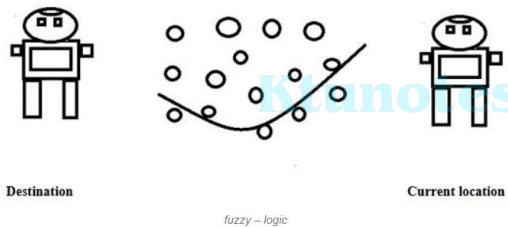
- ▶ There are three types of soft computing techniques which include the following.
- ▶ Artificial Neural Network
- ▶ Fuzzy Logic
- ▶ Genetic algorithm
- ▶ **Artificial Neural Network**



Hand written character recognition (Artificial Neural Networks)

Soft computing

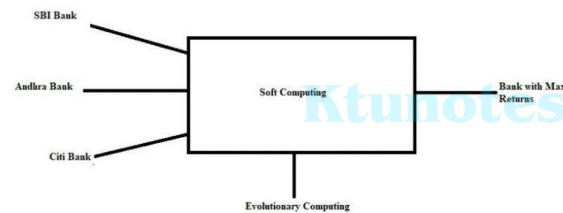
- Examples of soft computing-fuzzy logic



► An example of a robot that wants to move from one place to another within a short time where there are many obstacles on the way. Now the question arises is that how the robot can calculate its movement to reach the destination point, without colliding to any obstacle. These types of problems have uncertainty problem which can be solved using fuzzy logic

Soft computing

- Examples of soft computing-Genetic Algorithm



► A person who wants to invest some money in the bank, we know there are different banks available with different schemes and policies. Its individual interest how much amount to be invested in the bank, so that he can get maximum profit. There are certain criteria for the person that is, how he can invest and how can he get profited by investing in the bank. These criteria can be overcome by the "Evolutional Computing" algorithm like genetic computing.

The difference between hard computing and soft computing are as follows

Hard Computing	Soft Computing
• The analytical model required by hard computing must be precisely represented	• It is based on uncertainty, partial truth tolerant of imprecision and approximation.
• Computation time is more	• Computation time is less
• It depends on binary logic, numerical systems, crisp software.	• Based on approximation and dispositional.
• Sequential computation	• Parallel computation
• Gives exact output	• Gives appropriate output
• Examples: Traditional methods of computing using our personal computer.	• Example: Neural networks like Adaline, Madaline, ART networks, etc.

How Soft Computing?

- How a student learns from his teacher?
- Teacher asks questions and tell the answers then.
- Teacher puts questions and hints answers and asks whether the answers are correct or not.
- Student thus learn a topic and store in his memory.
- Based on the knowledge he solves new problems.
- This is the way how human brain works.
- Based on this concept Artificial Neural Network is used to solve problems.

How Soft Computing?

- ▶ **How a doctor treats his patient?**
- ▶ Doctor asks the patient about suffering.
- ▶ Doctor finds the symptoms of diseases.
- ▶ Doctor prescribed tests and medicines.
- ▶ This is exactly the way Fuzzy Logic works.
- ▶ Symptoms are correlated with diseases with uncertainty
- ▶ Doctor prescribes tests/medicines fuzzily

How Soft Computing?

- ▶ **How world selects the best?**
- ▶ It starts with a population (random).
- ▶ Reproduces another population (next generation).
- ▶ Rank the population and selects the superior individuals.
- ▶ Genetic algorithm is based on this natural phenomena.
- ▶ Population is synonymous to solutions.
- ▶ Selection of superior solution is synonymous to exploring the optimal solution

Neural Network

- ▶ Neural networks are a series of algorithms that mimic the operations of a human brain to recognize relationships between vast amounts of data.
- ▶ A neural network works similarly to **the human brain's neural network**.
- ▶ A "neuron" in a neural network is a mathematical function that collects and classifies information according to a specific architecture.
- ▶ Well suited for real time systems because of fast response and computation time(Parallel Architecture)

Artificial Neural Network

- ▶ ANN is an efficient Information processing paradigm inspired by biological nervous systems
- ▶ ANN is composed of large number of interconnected processing elements called nodes/neurons which operates in parallel
- ▶ ANN learn by example
- ▶ **Each neuron is connected with the other by a connection link.**
- ▶ Each connection link is associated with weights which contain information about the input signal.

Fuzzy Systems and Logic

- ▶ Concept is conceived by Lotfi Zadeh
- ▶ It is an organized method dealing with imprecise data
- ▶ Fuzzy logic is a super set of Boolean Logic
- ▶ Fuzzy set allows Partial Membership
- ▶ Multi valued Logic is allowed,
- ▶ Intermediate values between conventional evaluations such as YES/NO,T/F, Black/White etc

Genetic Algorithm

- ▶ genetic algorithms are inspired by Darwin's theory about evolution.
- ▶ Genetic Algorithm (GA) is a search based algorithms based on the principles of **Genetics and Natural Selection**.
- ▶ It is frequently used to find optimal or near-optimal solutions to difficult problems which otherwise would take a lifetime to solve.
- ▶ GAs are a subset of a much larger branch of computation known as **Evolutionary Computation**.
- ▶ Genetic algorithms are inspired by Darwin's theory about evolution.
- ▶ Genetic Algorithms (GAs) were invented by John Holland

Multi Objective Optimization & Hybrid Systems

- ▶ The optimal value or the best solution can be found through the optimization process.
- ▶ The optimization problems include looking for maximum or minimum value or using one objective or multi-objective.
- ▶ Problems that have more than one objective is referred to as multi-objective optimization (MOO).
- ▶ This type of problem is found in everyday life, such as mathematics, engineering, social studies, economics, agriculture, aviation, automotive, and many others.
- ▶ The MOO or the multi-objective optimization refers to finding the optimal solution values of more than one desired goals.
- ▶ The motivation of using the MOO is because in optimization, it does not require complicated equations, which consequently simplifies the problem.

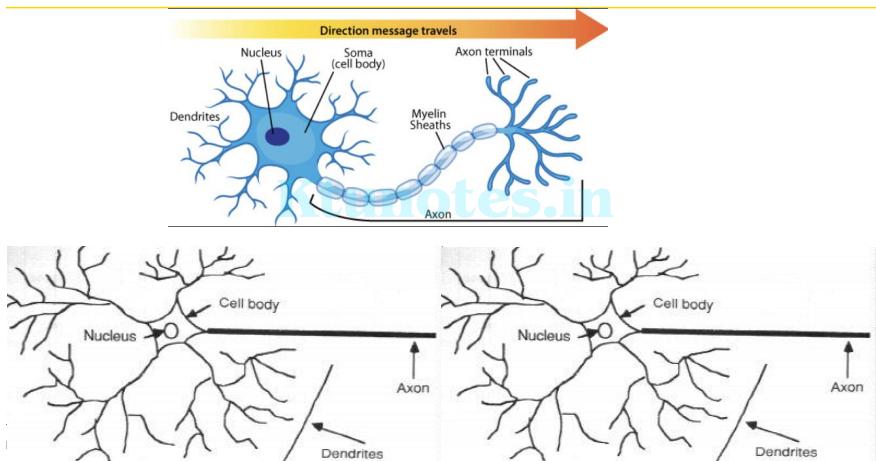
Applications of Soft Computing

- ▶ Handwriting Recognition
- ▶ Image Processing and Data Compression
- ▶ Automotive Systems and Manufacturing
- ▶ Soft Computing to Architecture
- ▶ Decision-support Systems
- ▶ Soft Computing to Power Systems
- ▶ Neuro Fuzzy systems
- ▶ Fuzzy Logic Control
- ▶ Machine Learning Applications
- ▶ Speech and Vision Recognition Systems

About Human Brain

- Cells within the nervous system, called **neurons**, communicate with each other in unique ways ,which doesn't regenerate.
- **The neuron is the basic working unit of the brain**, a specialized cell designed to transmit information to other nerve cells, muscle, or gland cells.
- The human brain contains around 100 billion neurons(**interconnected neurons**)and each neuron can connect up to 2,00,000 other neurons.(10^{11} neurons approximately)
- Each neuron consists of a cell body, dendrites, and an axon.
- The approximate size of neuron body(in micrometer) is 10-80. The gap of Biological neuron at synapses is 200nm.
- A synapse is the connection between nodes, or neurons, in an artificial neural network (ANN).
- The power of human brain comes from this sheer number of neurons and their multiple interconnections

Biological Neurons



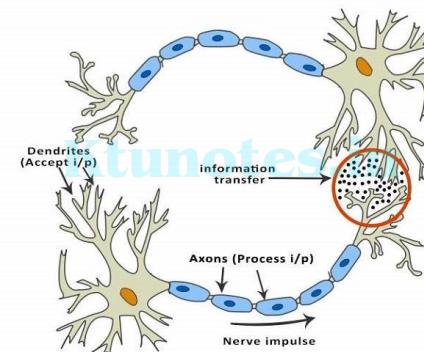
Biological Neurons

- A neuron is the basic processing unit in a neural network sitting on our brain.
- It consists of
 - Soma or cell body where cell nucleus is situated
 - Dendrites -Tree like network made of nerve fibre connected to cell body which **receives signals**
 - Axon-Single long connection extending from the cell body which carries impulses of the neuron and **transmit signals**
- Synapse -The end of the **axon splits into fine strands, each strand terminates into a small bulb like organ called Synapse.**
- There are approximately 10^4 synapses/neurons in human brain.There are 10^{15} synaptic connection are there in human brain

Biological Neurons

- ▶ Neuron collects signals from **dendrites**.
- ▶ Sends out spikes of electrical activity through an **axon**, which splits into thousands of branches.
- ▶ Electric pulses are passed between the synapse and the dendrites
- ▶ At the end, a **synapses** converts activity into either excitatory or inhibiting activity of a dendrite at another neuron.
- ▶ **Neuron fires during an excitatory activity and surpasses inhibitory activity**
- ▶ If the electrical potential inside the body of the receiving cell reaches a threshold, then the receiving cell fires and pulse or action of fixed strength and duration is sent out through the axon to the synaptic junctions of the other cell.
- ▶ After firing the cell has to wait for a period called refractory period before it can fire again.

Biological Neurons

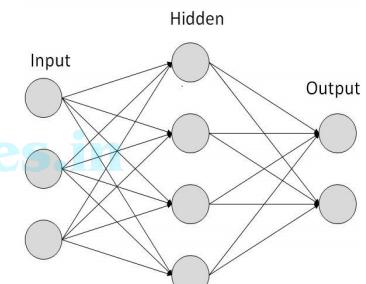


Artificial Neural Network

- ▶ ANN is an efficient Information processing paradigm which resembles the characteristics of a biological neural network.
- ▶ ANN is composed of **large number of interconnected processing elements called neurons** connected by synapses which operates in parallel
- ▶ Each neuron is connected with the other by a connection link.
- ▶ Each **connection link is associated with weights** which contain information about the input signal.
- ▶ This information is used by the neurons net to solve a particular problem
- ▶ Each neuron has an individual state of its own. This state can be varied based on the **activation signal**.
- ▶ The activation signal of a neuron is transmitted to other neurons
- ▶ A neuron can send only one signal at a time ,which can be transmitted to other neurons.

Artificial Neural Network

- ▶ ANN is an efficient Information processing paradigm which resembles the characteristics of a biological neural network.
- ▶ ANN is composed of **large number of interconnected processing elements called neurons** connected by synapses which operates in parallel
- ▶ ANN is a model that simulates the human brain created artificially to solve complex problems



Artificial Neuron Model

- ▶ consider a set of neurons, say X_1 & X_2 , the input signals x_1, x_2 and Y_{in} is the Net input and Y is the output neuron which receives the signal.
- ▶ Each neuron is connected to other neuron by means of directed communication links and associated with weights w_1, w_2

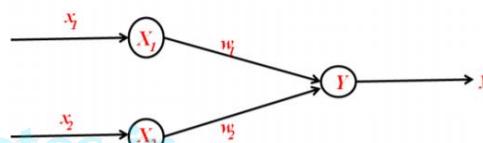
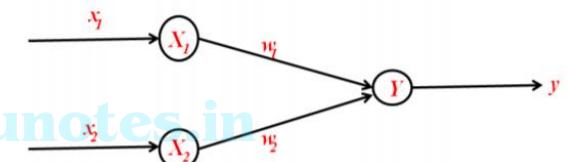


Figure 1.1: Architecture of a simple artificial neural net

Artificial Neuron Model

- ▶ consider a set of neurons, say X_1, X_2 which are the input signals and
- ▶ Y_{in} is the Net input and
- ▶ ' y ' is the output of neural net, which receives the signal.
- ▶ Each neuron is connected to other neuron by means of directed communication links and associate with weights w_1, w_2



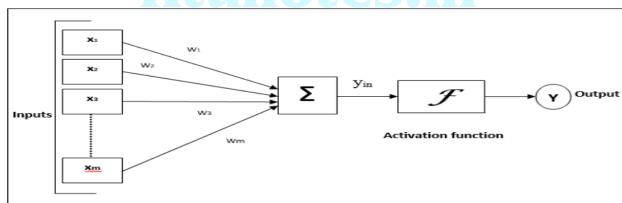
$$\text{Net input, } y_{in} = x_1 w_1 + x_2 w_2$$

$$y = f(y_{in})$$

Output = Function(net input calculated)

Artificial Neuron Model

- ▶ ANN is a model that simulates the human brain created artificially to solve complex problems
- ▶ To depict the basic operation of neuron net ,consider a set of neurons, say x_1, x_2, \dots, x_m are the input signals and Y_{in} is the Net input and ' y ' is the output neuron which receives the signal.
- ▶ Each neuron is connected to other neuron by means of directed communication links and associated with weights w_1, w_2, \dots, w_m



Artificial Neuron Model

For the above general model of artificial neural network, the net input can be calculated as follows -

$$y_{in} = x_1 \cdot w_1 + x_2 \cdot w_2 + x_3 \cdot w_3 \dots x_m \cdot w_m$$

$$\text{i.e., Net input } y_{in} = \sum_i^m x_i \cdot w_i$$

The output can be calculated by applying the activation function over the net input.

$$Y = F(y_{in})$$

Output = function netinputcalculated

Similarities of ANN and BNN

Biological Neural Network BNN	Artificial Neural Network ANN
Soma	Node
Dendrites	Input
Synapse	Weights or Interconnections
Axon	Output

Comparison between ANN and BNN

	BNN	ANN
Processing	Massively parallel, slow but superior than ANN	Massively parallel, fast but inferior than BNN
Size	10^{11} neurons and 10^{15} interconnections	10^2 to 10^4 nodes
Learning	They can tolerate ambiguity	Very precise, structured and formatted data is required to tolerate ambiguity
Fault tolerance	Performance degrades with even partial damage	It is capable of robust performance, hence has the potential to be fault tolerant
Storage capacity	Stores the information in the synapse	Stores the information in continuous memory locations

Advantages of ANN

- ▶ Adaptive learning: An ability to learn how to do tasks based on the data given for training or initial experience.
- ▶ Self-Organisation: An ANN can create its own organisation or representation of the information it receives during learning time.
- ▶ Real Time Operation: ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.
- ▶ Fault Tolerance via Redundant Information Coding:

Features of ANN

- ▶ Parallel Distributed information processing
- ▶ High degree of connectivity between basic units
- ▶ Connections are modifiable based on experience
- ▶ Learning is a continuous unsupervised process
- ▶ Learns based on local information
- ▶ Performance degrades with less units

Problem

Q1: Calculate the net input of the following network

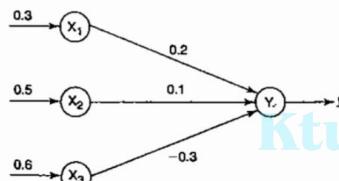


Figure 1 Neural net.

The given neural net has 3 I/P neurons and one O/P neuron.
 Inputs $x = [x_1, x_2, x_3] = [0.3, 0.5, 0.6]$
 weights, $W = [w_1, w_2, w_3] = [0.2, 0.1, -0.3]$
 The net i/p can be calculated as

$$Y_{in} = x_1 w_1 + x_2 w_2 + x_3 w_3$$

$$\begin{aligned} &= 0.3 \times 0.2 + 0.5 \times 0.1 + 0.6 \times -0.3 \\ &= \underline{\underline{-0.07}} \end{aligned}$$

Artificial Neuron Model- additional points

- The function to be applied over the net input is called the activation functions
- $i \rightarrow$ represent the i th processing element
- $w \rightarrow$ weight represents the strength of synapse connecting the input and output neurons
- Excitatory Synapse -Positive Weights
- Inhibitory Synapse -Negative Weights
- Bias \rightarrow Bias included in the network has its impact in calculating the net input
- Bias is included by adding $x_0 = 1$ to the input vector x . so the input signal value can be 1 and denoted by b and can be considered like another weight with weight 1.
- Positive bias helps in increasing the net input of the network
- Negative bias helps in decreasing the net input of the network
- As a result of the bias effect ,output of the network can be varied

Problem

Q2: Calculate the net input of the following network

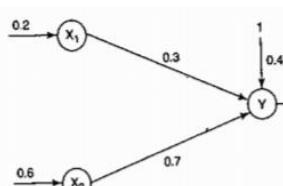


Figure 2 Simple neural net.

The net input to the output neuron is

$$Y_{in} = b + \sum_{i=1}^n x_i w_i$$

Input, $x = [x_1, x_2] = [0.2, 0.6]$
 weight, $W = [w_1, w_2] = [0.3, 0.7]$
 The net i/p calculated with bias included,

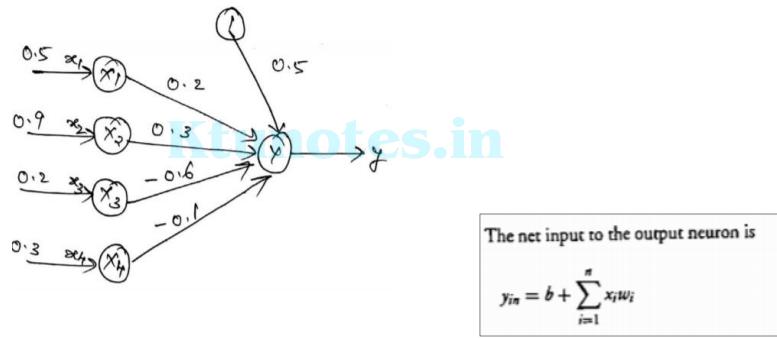
$$Y_{in} = x_0 w_0 + x_1 w_1 + x_2 w_2$$

where x_0 is the bias i/p and $w_0(0)$ is the bias weight

$$\begin{aligned} \text{as, } x_0 &= 1 \text{ and } w_0 \text{ or } b = 0.45 \\ \therefore Y_{in} &= 1 \times 0.45 + 0.2 \times 0.3 + 0.6 \times 0.7 \\ &= \underline{\underline{0.93}} \end{aligned}$$

Problems

Q3: Calculate the net input of the following network



The given neural net has 4 i/p neurons, and one o/p neuron.

Input, $x = [x_1, x_2, x_3, x_4] = [0.5, 0.9, 0.2, 0.3]$
 weight, $w = [w_1, w_2, w_3, w_4] = [0.2, 0.3, -0.6, -0.1]$

The net r/p calculated with bias included

$$Y_{in} = x_0 w_0 + x_1 w_1 + x_2 w_2 + x_3 w_3 + x_4 w_4$$

where x_0 is the bias i/p and w_0 is the bias weight.

$$\text{Given, } x_0 = 1 \text{ and } w_0 = 0.5 \\ \therefore Y_{in} = 1 \times 0.5 + 0.5 \times 0.2 + 0.9 \times 0.3 \\ + 0.2 \times -0.6 + 0.3 \times -0.1 \\ = \underline{\underline{0.72}}$$

Basic models of Artificial Neural Networks

Basic models of artificial neural networks

- The models of Artificial neural network is characterized by **three basic entities** namely
 1. *connections* between the neurons called its *architecture*
 2. *training or learning rules adopted for updating and adjusting weights* on the connections
 3. Its *internal state* called its *Activation function*.
- The arrangement of neurons into layers and the connection patterns within and between layers is called the net *architecture*

1.Connections

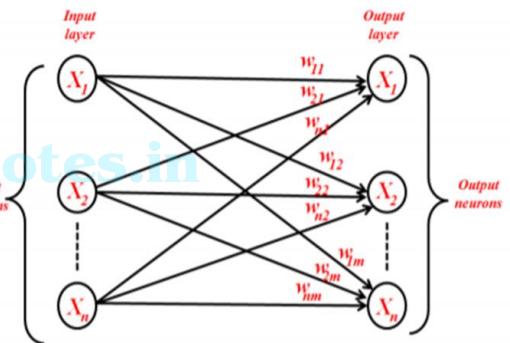
- Generally there are two category of Neural network Connections
 - Feed Forward/Recurrent
 - Single Layer/Multi Layer
- Connections across the layers in standard topologies can be in feedforward manner or in feedback manner but not both.
- A neural net in which the signals flow from the input units to the output units in a forward direction is called feed forward nets.
- Interconnected competitive net in which there are closed loop signal paths from a unit back to it is called a recurrent network.

1.Connections

- An ANN consists of set of highly interconnected processing elements(Neurons)
- The point where the connection originates and terminated should be noted and functions of each processing element of an ANN is specified.
- Connection means the network topology ,which is the arrangement of a network along with its nodes and connecting lines.
- There are five types of Neural Network Connections
 - Single layer feed forward network
 - Multilayer feed forward network
 - Single node with its own feed back
 - Single layer recurrent network
 - Multilayer recurrent network

1.Connections Single layer feed forward network

- A neural net with only input layer and output layer is called single layer neural network.
- The i/p layer and o/p layer are linked with each other
- The input layer receives input data from the user
- The output layer sends its calculated output to the user from which decision can be made.

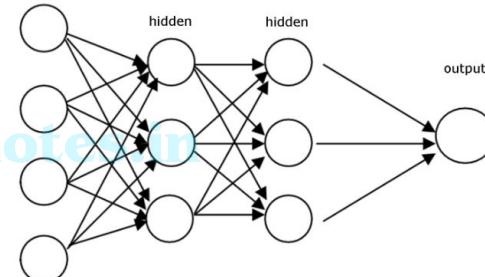


1.Connections Multi layer feed forward network

- A neural network with input layer, one or more hidden layers and an output layer is called a multilayer neural network.

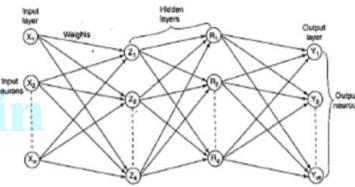
The feed forward model is the simplest form of neural network as information is only processed in one direction.

While the data may pass through multiple hidden nodes, it always moves in one direction and never backwards.



1.Connections Multi layer feed forward network

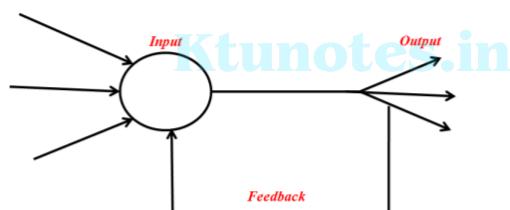
- A neural network with input layer, one or more hidden layers and an output layer is called a multilayer neural network.
- input layer -The input layer receives input data from the user and propagates a signal to the next layer called the hidden layer. While doing so it multiplies the weight along with the input signal.
- Hidden layer-Any layer which lies between the input and the output layers is called hidden layer. The hidden layer is internal to the network and has no direct contact with the external environment. The hidden layer performs all kinds of computation on the features entered through the input layer and transfers the result to the output layer.
- Output layer-The output layer sends its calculated output to the user from which decision can be made



1.Connections

Single node with its own feed back

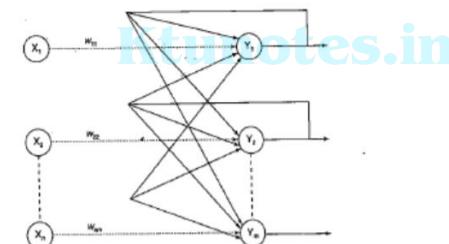
- This shows a simple recurrent neural network model having a single neuron with Feedback to itself



1.Connections

Single layer recurrent network

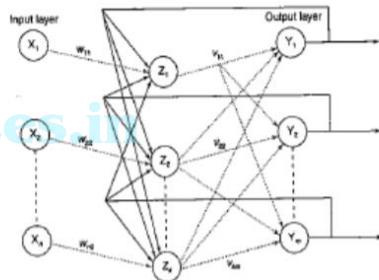
- Single layer network with a feedback connection in which a Processing elements output can be directed back to PE itself or to the other PE or to both



1.Connections

Multi layer recurrent network

- ▶ A PE's output can be directed back to the nodes in a preceding layer forming a multilayer recurrent network
- ▶ A PE's o/p can be directed back to the PE's itself and to PE in the same layer or preceding layer.



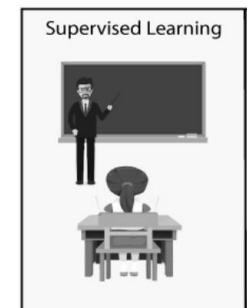
Training /Learning
Ktunotes.in

2.Learning

- ▶ Learning, in artificial neural network, is the method of modifying the weights of connections between the neurons of a specified network.
- ▶ The main property of ANN is its ability to learn
- ▶ Generating the output in response to the input after processing is called **Learning**.
- ▶ The method of setting the values of the weights called **training** is an important characteristic of neural nets.
- ▶ Based on the training methodology used neural nets can be distinguished into **supervised or unsupervised** neural nets
- ▶ Types of Learning
 - ▶ Supervised Learning
 - ▶ Unsupervised Learning
 - ▶ Reinforcement Learning

2.Learning

- ▶ **supervised learning** takes place under the supervision of a teacher. This learning process is dependent.
- ▶ Basically supervised learning is when we teach or train the machine **using data that is well labelled**. Which means some data is already tagged with the correct answer.
- ▶ After that, the machine is provided with a new set of examples(data) so that the supervised learning algorithm **analyses the training data**(set of training examples) and **produces a correct outcome** from labelled data



2.Learning-**supervised** learning example

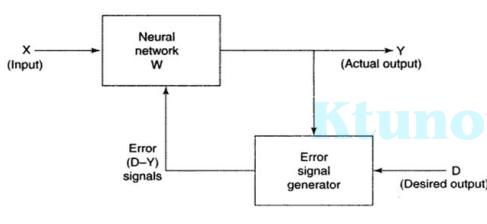
- ▶ suppose you are given a basket filled with different kinds of fruits.
- ▶ The first step is to train the machine with all the different fruits one by one like this
- ▶ If the shape of the object is rounded and has a depression at the top, is red in color or green, then it will be labeled as -**Apple**.
- ▶ If the shape of the object is a long curving cylinder having Green-Yellow color, then it will be labeled as -**Banana**.
- ▶ Now suppose after training the data, you have given a new separate fruit, say Banana from the basket, and asked to identify it.



2.Learning- Supervised Learning

- ▶ During the **training** of ANN under supervised learning, the **input vector is presented to the network**, which will produce an output vector. This **output vector is compared with the desired/target output vector**.
- ▶ An error signal is generated if there is a difference between the actual output and the desired/target output vector. On the basis of this error signal, the **weights would be adjusted** until the actual output is matched with the desired output.
- ▶ In ANN each input vector requires a corresponding target vector ,which represents the desired output.The **i/p vector along with target vector is called Training Pair**.
- ▶ In this type a supervisor is required for error minimization.

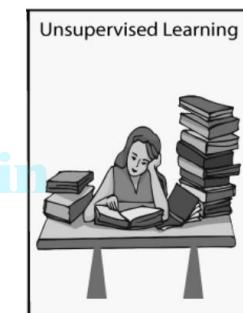
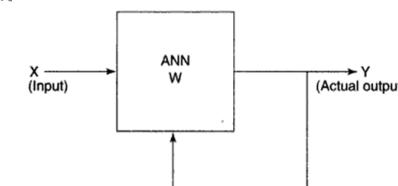
2.Learning- Supervised Learning



- ▶ Supervised learning is classified into two categories of algorithms:
- ▶ **Classification:** A classification problem is when the output variable is a category,
- ▶ **Regression:** A regression problem is when the output variable is a real value,
- ▶ Supervised learning deals with or learns with "labeled" data.
- ▶ This implies that some data is already tagged with the correct answer.

2.Learning Unsupervised Learning

- ▶ In unsupervised learning **no teacher** is available.
- ▶ Learning process is independent.
- ▶ The learner only discovers persistent patterns in the data consisting of a collection of perceptions. This is also called explorat



2.Learning

Unsupervised Learning

- ▶ The i/p vectors of similar types are grouped without the use of training data.
- ▶ In the training process ,the network receives the i/p patterns and organizes these patterns to form **clusters**.
- ▶ When a new input pattern is applied the neural network gives the output response indicating the class to which the i/p pattern belongs .
- ▶ If for an i/p ,a pattern class cannot be found ,then a new class can be generated.
- ▶ By **self organizing process exact clusters** will be formed by discovering **similarities and dissimilarities** among the objects
- ▶ Finding out malicious network attacks from a sequence of anomalous data packets is an example of unsupervised learning.

2.Learning

Unsupervised Learning

- ▶ Unlike supervised learning, no teacher is provided that means no training will be given to the machine.
- ▶ Therefore the machine is restricted to find the hidden structure in unlabeled data by itself.
- ▶ Here the task of the machine is to group unsorted information according to similarities, patterns, and differences without any prior training of data.
- ▶ Ex: the machine has no idea about the features of dogs and cats so we can't categorize it as 'dogs and cats'. But it can categorize them according to their similarities, patterns, and differences,
- ▶ It mainly deals with unlabelled data.



2.Learning

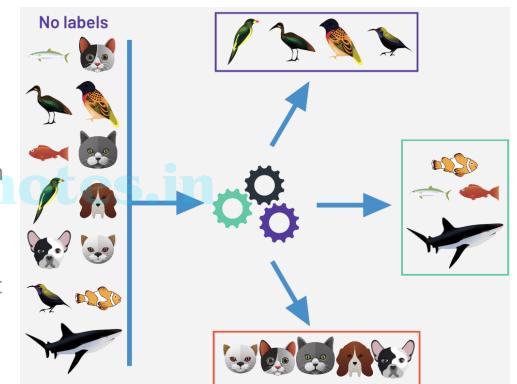
Unsupervised Learning

- ▶ The i/p vectors of similar types are grouped without the use of training data.
- ▶ In the training process ,the network receives the i/p patterns and organizes these patterns to form **clusters**.
- ▶ When a new input pattern is applied the neural network gives the output response indicating the class to which the i/p pattern belongs .
- ▶ If for an i/p ,a pattern class cannot be found ,then a new class can be generated.
- ▶ By **self organizing process exact clusters** will be formed by discovering **similarities and dissimilarities** among the objects
- ▶ Finding out malicious network attacks from a sequence of anomalous data packets is an example of unsupervised learning.

2.Learning

Unsupervised Learning

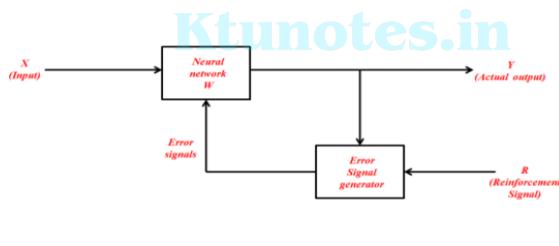
- ▶ Unsupervised learning is classified into two categories of algorithms:
- ▶ **Clustering:** A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.
- ▶ **Association:** An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.



2.Learning

Reinforcement Learning

- ▶ Similar to supervised learning. But only critic information available.
- ▶ In reinforcement learning the learning is **based on critic information** and the feedback sent is known as Reinforcement signal.
- ▶ The network receives some feedback from the environment.

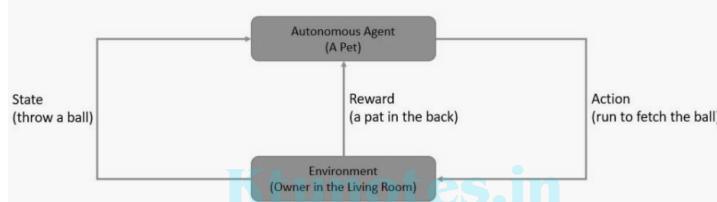


2.Learning

Reinforcement Learning

- ▶ The ANN makes a decision by observing its environment.
- ▶ If the observation is negative, the network adjusts its weights to be able to make a different required decision the next time..
- ▶ Examples include a robot in a unknown terrain where its get a punishment when its hits an obstacle and reward when it moves smoothly.
- ▶ It is a trial and error method.
- ▶ Reinforcement learning is about an autonomous agent taking suitable actions to maximize rewards in a particular environment. Over time, **the agent learns from its experiences and tries to adopt the best possible behavior**.
- ▶ In the case of reinforcement learning, we limit human interaction to changing the environment states, and the system of rewards and penalties. This **setup is known as the Markov Decision Process**

2.Learning Reinforcement Learning



- ▶ example of training a pet.
- ▶ consider the pet as an autonomous agent. If you're trying to train the pet in our living room, that can be considered as the environment:
- ▶ you can throw a ball and expect the pet to run and fetch it. Here, throwing the ball represents a state that the environment presents, and running to fetch it represents an action that the pet may take.
- ▶ issue rewards immediately or delay them to some point in the future.

Activation Function
Ktunotes.in

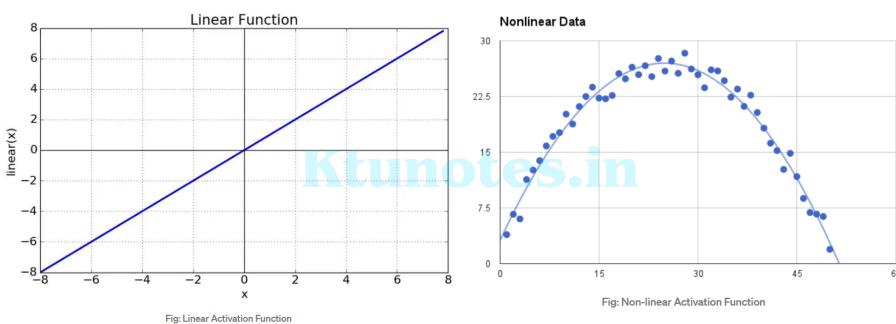
3. Activation Function

- ▶ In ANN activation function may be defined as the extra force or effort applied over the net input to obtain an exact output. Activation function is applied over the net input to calculate the o/p of the ANN
- ▶ An Activation Function decides whether a neuron should be activated or not by calculating weighted sum and further adding bias with it.
- ▶ This means that it will decide whether the neuron's input to the network is important or not in the process of prediction using simpler mathematical operations.
- ▶ Linear and Non linear activation functions can be used to generate the neurons response.
- ▶ Linear
 - ▶ Identity functions
 - ▶ Binary Step Function
 - ▶ Bipolar step functions
- ▶ Non Linear
 - ▶ Sigmoid Function
 - ▶ Ramp Function

3. Activation Function

- ▶ linear activation function
 - ▶ It is simply a linear regression model.
 - ▶ It has limited power and ability to handle complexity varying parameters of input data.
 - ▶ with linear activation functions, no matter how many layers in the neural network, the last layer will be a linear function of the first layer (because a linear combination of linear functions is still a linear function).
 - ▶ So a linear activation function turns the neural network into just one layer.
- ▶ Non-Linear Activation Functions
 - ▶ Modern neural network models use non-linear activation functions.
 - ▶ They allow the model to create complex mappings between the network's inputs and outputs, which are essential for learning and modeling complex data, such as images, video, audio, and data sets which are non-linear or have high dimensionality.

3. Activation Function

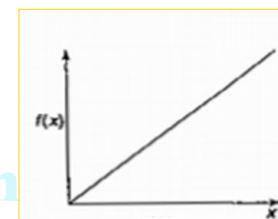


3. Activation Function

1. **Identity function:** It is a linear function and can be defined as

$$f(x) = x \quad \text{for all } x$$

The output here remains the same as input.
The input layer uses the identity activation function.



- ▶ Two major problems:
 - ▶ Back-propagation is not possible — The derivative of the function is a constant, and has no relation to the input, X. So it's not possible to go back and understand which weights in the input neurons can provide a better prediction.
 - ▶ All layers of the neural network collapse into one — with linear activation functions, no matter how many layers in the neural network, the last layer will be a linear function of the first layer

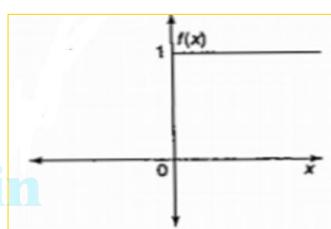
3. Activation Function

2. Binary step function: This function can be defined as

$$f(x) = \begin{cases} 1 & \text{if } x \geq \theta \\ 0 & \text{if } x < \theta \end{cases}$$

where θ represents the threshold value.

This function is most widely used in single-layer nets to convert the net input to an output that is a binary (1 or 0).



limitations of binary step function:

- It cannot provide multi-value outputs so, it cannot be used for multi-class classification problems.
- The gradient of the step function is zero, which causes a hindrance in the backpropagation process.

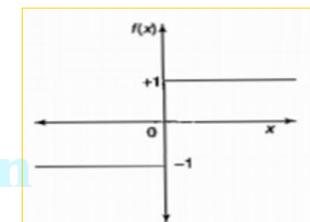
3. Activation Function

3. Bipolar step function: This function can be defined as

$$f(x) = \begin{cases} 1 & \text{if } x \geq \theta \\ -1 & \text{if } x < \theta \end{cases}$$

where θ represents the threshold value.

This function is also used in single-layer nets to convert the net input to an output that is bipolar (+1 or -1).



► In the Bipolar Step Function, if the value of Y is above a certain value known as the threshold, the output is +1 and if it's less than the threshold then the output is -1.

► It has bipolar outputs (+1 to -1). It can be utilized in single-layer networks.

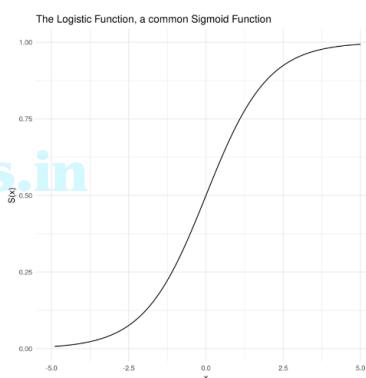
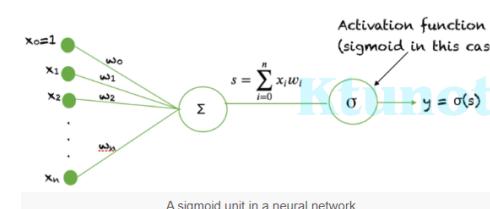
3. Activation function

4. Sigmoidal Functions

- A sigmoidal function is a mathematical function having a characteristic "S"-shaped curve or sigmoid curve.
- When the activation function for a neuron is a sigmoid function it is a guarantee that the output of this unit will always be between 0 and 1.
- Also, as the sigmoid is a non-linear function, the output of this unit would be a non-linear function of the weighted sum of inputs.
- This function takes any real value as input and outputs values in the range of 0 to 1.
- The larger the input (more positive), the closer the output value will be to 1.0, whereas the smaller the input (more negative), the closer the output will be to 0.0.
- They are widely used in backpropagation networks because of the relationship between the value of the functions at a point and the value of the derivative at that point which reduces the computational burden during training.
- Sigmoidal functions are of two types
 - Binary Sigmoidal Function
 - Bipolar Sigmoidal function

3. Activation function

Sigmoidal Functions



3. Activation Function

Binary sigmoid function:

It is also termed as logistic sigmoid function or unipolar sigmoid function.

It can be defined as

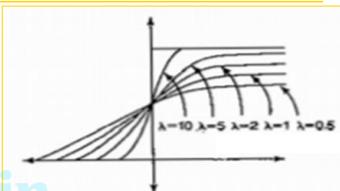
$$f(x) = \frac{1}{1 + e^{-\lambda x}}$$

where λ is the steepness parameter.

The derivative of this function is

$$f'(x) = \lambda f(x)[1 - f(x)]$$

Here the range of the sigmoid function is from 0 to 1.



- Binary Sigmoid Function or **Sigmoid function** is a logistic function where the output values are either binary or vary from 0 to 1.
- It is differentiable, non-linear, and produces non-binary activations
- sigmoid activation is not a zero-centric function.

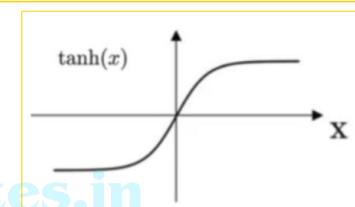
3. Activation Function

Bipolar sigmoid function:

This function is defined as

$$f(x) = \frac{2}{1 + e^{-\lambda x}} - 1$$

$$= \frac{1 - e^{-\lambda x}}{1 + e^{-\lambda x}}$$



► Hyperbolic Tangent Function or Tanh is a logistic function where the output value varies from -1 to 1. Also known as **Bipolar Sigmoid Function**.

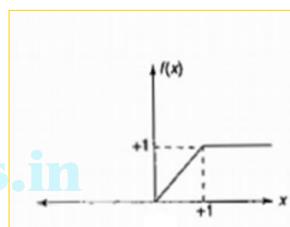
► The output of Tanh centers around 0 and sigmoid's around 0.5. Tanh Convergence is usually faster if the average of each input variable over the training set is close to zero

3. Activation Function

5. Ramp function:

The ramp function is defined as

$$f(x) = \begin{cases} 1 & \text{if } x > 1 \\ x & \text{if } 0 \leq x \leq 1 \\ 0 & \text{if } x < 0 \end{cases}$$



- ReLU stands for the rectified linear unit (ReLU). It is the most used activation function in the world. It output 0 for negative values of x. This is also known as a ramp function..
- The name of the **ramp function** is derived from the appearance of its graph.
- ReLU(Rectified Linear Unit) is like a linearity switch. If you don't need it, you "switch" it off. If you need it, you "switch" it on.
- ReLU also provides the benefit of sparsity and sigmoids result in dense representations. Sparse representations are more useful than dense representations.

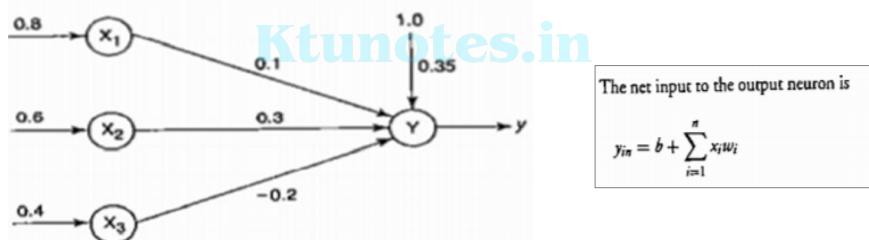
Problems
Ktunotes.in

Q1 :Activation Function

Obtain the output of the neuron Y for the network shown in Figure using the following activation functions

i)Binary Sigmoidal

ii)Bipolar sigmoidal



Q1 :Activation Function

The given network has 3 i/p neurons and one o/p neuron.

Input, $X = [x_1, x_2, x_3] = [0.8, 0.6, 0.4]$

Weight, $W = [w_1, w_2, w_3] = [0.1, 0.3, -0.2]$

Where x_0 is the bias i/p and w_0 is the bias weight i.e., $x_0 = 1$ and $w_0 = 0.35$

$$\begin{aligned} \therefore y_{in} &= x_0 w_0 + x_1 w_1 + x_2 w_2 + x_3 w_3 \\ &= 1 \times 0.35 + 0.8 \times 0.1 + 0.6 \times 0.3 + 0.4 \times -0.2 \\ &= \underline{\underline{0.53}} \end{aligned}$$

Q1 :Activation Function

(i) For binary sigmoidal activation function,

$$f = f(y_{in}) = \frac{1}{1 + e^{-y_{in}}}$$

$$= \frac{1}{1 + e^{-0.53}}$$

$$= \underline{\underline{0.625}}$$

Q1 :Activation Function

(ii) For bipolar sigmoidal activation function

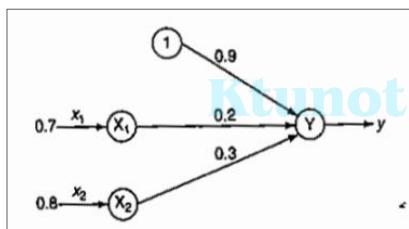
$$f = f(y_{in}) = \frac{2}{1 + e^{-y_{in}}} - 1$$

$$= \frac{2}{1 + e^{-0.53}} - 1$$

$$= \underline{\underline{0.259}}$$

Q2 :Activation Function

- Calculate the net o/p of the NN using the bipolar and binary sigmoidal activation function



The net input to the output neuron is

$$y_{in} = b + \sum_{i=1}^n x_i w_i$$

Q1 :Activation Function

The given n/w has 2 i/p neurons and one o/p neuron.

Input, $x = [x_1, x_2] = [0.7, 0.8]$

Weight, $w = [w_1, w_2] = [0.2, 0.3]$

The net i/p calculated with bias included,

$$y_{in} = x_0 w_0 + x_1 w_1 + x_2 w_2$$

where x_0 is the bias i/p and w_0 is the bias weight
i.e., $x_0 = 1$, $w_0 = 0.9$

$$\therefore y_{in} = 1 \times 0.9 + 0.7 \times 0.2 + 0.8 \times 0.3$$

$$= 1.28$$

Q1 :Activation Function

(1) for binary sigmoidal activation function

$$f = f(y_{in}) = \frac{1}{1 + e^{-y_{in}}}$$

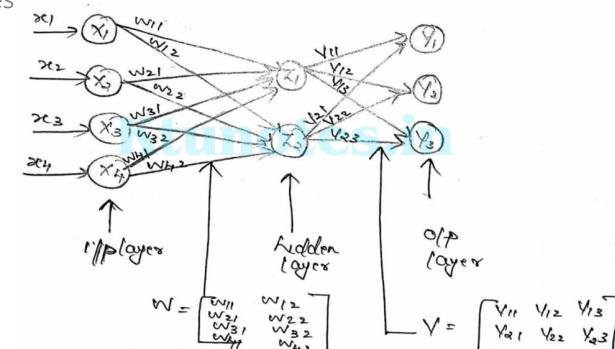
$$= \frac{1}{1 + e^{-1.28}} = 0.78$$

(2) for bipolar sigmoidal activation function

$$f = f(y_{in}) = \frac{2}{1 + e^{-y_{in}}} - 1 = \frac{2}{1 + e^{-1.28}} - 1 = 0.56$$

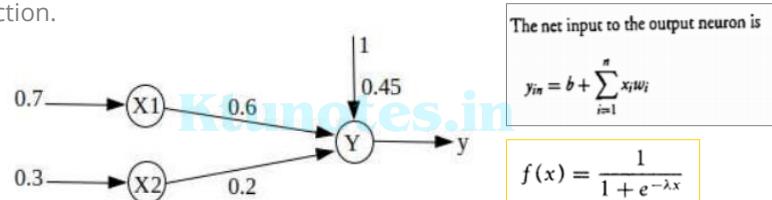
Q3 :Activation Function

- Construct a feed forward network with 4 input nodes, 2 hidden nodes and 3 o/p nodes



Q4 :Activation Function [Univ ques January 2022]

- Calculate the net input to the neuron Y for the network shown in figure. Compute output of the neuron Y using binary sigmoidal activation function.

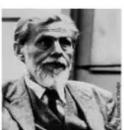


► Net input to the neuron Y = 0.93

► Output of the neuron Y using binary sigmoidal activation function = 0.72

McCulloch and Pitts Neuron

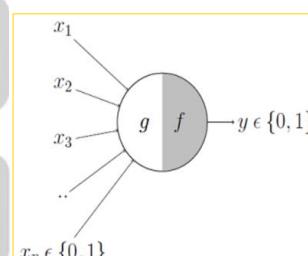
McCulloch and Pitts Neuron



Warren McCulloch was a neuroscientist who created computational models based on threshold logic which split the inquiry into two distinct approaches, focused on biological processes in the brain and application of neural networks to artificial intelligence.

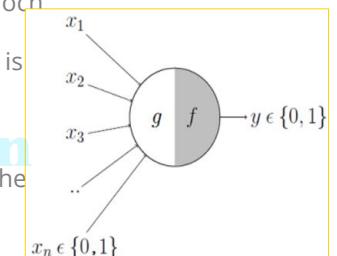


Walter Pitts was a logician who proposed the first mathematical model of a neural network. The unit of this model, a simple formalized neuron, is still the standard of reference in the field of neural networks. It is often called a McCulloch-Pitts neuron.

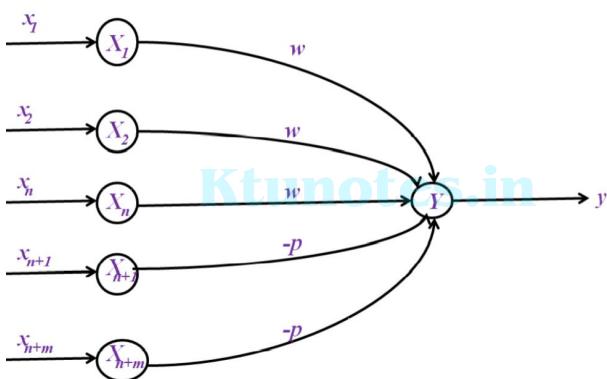


McCulloch and Pitts Neuron

- The first Mathematical /computational model of a Biological Neuron was proposed by Warren MuCulloch (neuroscientist) and Walter Pitts (logician) in 1943.
- It is considered as the earliest Neural Network and is also called **M-P Neuron**.
- M P Neurons are connected by directed weighted paths.
- Its main purpose is to **process the binary data**. i-e the activation of a M-P neuron is binary(at any time neuron may fire/may not fire).
- Logic gates /functions can be used in application
- It may be divided into 2 parts. The first part, **g takes an input**, performs **an aggregation** and based on the aggregated value the second part, **f makes a decision**.



Architecture of McCulloch and Pitts Neuron



Architecture of McCulloch and Pitts Neuron

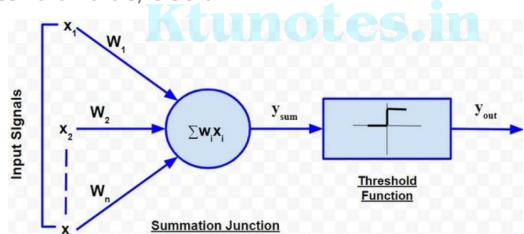
- Weights associated with the communication links excitatory(+ve weights) or inhibitory(-ve)
- M-P Neuron has either excitatory or inhibitory connections
- i/p's from x_1 to x_n are excitatory weighted connections and will have same weights ($w > 0$) and i/p's from x_{n+1} to x_{n+m} are inhibitory
- There is a fixed threshold for each neuron.
- Firing of Neuron is based upon a Threshold.

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

- For inhibition to be abs $\theta > nw - p$ threshold with the activation function should satisfy the following condition.
- Where $n \rightarrow$ no of neurons , $w \rightarrow$ weight of excitatory connections , $p \rightarrow$ weight of inhibitory connections
- The o/p will fire if it cor $kw \geq \theta > (k - 1)w$ excitatory inputs but no inhibitory inputs where

Architecture of McCulloch and Pitts Neuron

- Simple McCulloch-Pitts neurons can be used to design logical operations.
- For that purpose, the connection weights need to be correctly decided along with the threshold function
- The inputs of the McCulloch-Pitts neuron could be either 0 or 1. It has a threshold function as an activation function. So, the output signal y_{out} is 1 if the input y_{sum} is greater than or equal to a given threshold value, else 0



Example of McCulloch and Pitts Neuron

- Person X carries an umbrella if it is sunny or if it is raining. We need to decide when Person X will carry the umbrella
- There are four given situations. 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 analyze the situations using the McCulloch-Pitts neural model, consider the input signals as follows:
 - X_1 : Is it raining? [0/1]
 - X_2 : Is it sunny? [0/1]
- So, the value of both scenarios can be either 0 or 1..

Example of McCulloch and Pitts Neuron

- ▶ Person X carries an umbrella if it is sunny or if it is raining. We need to decide when Person X will carry the umbrella
 - ▶ 1: not raining, not sunny
 - ▶ 2: not raining , sunny
 - ▶ 3: raining, and not sunny
 - ▶ 4: raining as well as sunny
- ▶ From the truth table, we can conclude that in the situations where the value of y_{out} is 1 Person X needs to carry an umbrella.
- ▶ We can use the value of both weights X_1 and X_2 as 1 and a threshold function as 1
- ▶ Hence, he will need to carry an umbrella in scenarios 2, 3 and 4.

Situation	x1	x2	ysum	yout
1	0	0	0	0
2	0	1	1	1
3	1	0	1	1
4	1	1	2	1

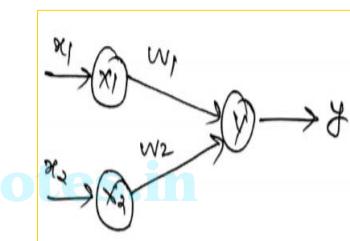
$$y_{out} = f(y_{sum}) = \begin{cases} 1, & x \geq 1 \\ 0, & x < 1 \end{cases}$$

Problems McCulloch and Pitts Neuron

Implement AND function using M-P Neuron Model(Use Binary Data)

Implement AND function using M-P Neuron Model(Use Binary Data)

Truth table of AND		
x ₁	x ₂	y
0	0	0
0	1	0
1	0	0
1	1	1



From the truth table, it is clear that for AND function, the op neuron will only fire if $x_1 = 1$ and $x_2 = 1$.

Implement AND function using M-P Neuron Model(Use Binary Data)

Case 1:- Assume that all the weights are excitatory, $w_1 = w_2 = 1$.
The net i/p calculated for four i/p

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

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

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

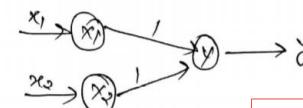
$$(1,1) \Rightarrow y_{in} = x_1 w_1 + x_2 w_2 \\ = 1 \times 1 + 1 \times 1 = 2$$

$$\theta > nw - p \\ n=2 \\ w=1 \\ p=0 \\ \text{so } nw-p=2 \times 1 - 0 \\ = 2$$

Implement AND function using M-P Neuron Model(Use Binary Data)

$x_1 = 1$ and $x_2 = 1$ (whose $y_{in} = 2$), we need to compare its y_{in} with others to set threshold value.

From the y_{in} calculated, it is clear that if we set threshold, $\theta \geq 2$, then truth-table is satisfied, with $w_1 = 1$ and $w_2 = 1$. Thus the final net o/p, $y = f(y_{in})$



$$y = f(y_{in})$$

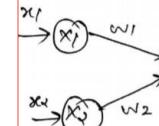
$$= \begin{cases} 1 & \text{if } y_{in} \geq 2 \\ 0 & \text{if } y_{in} < 2 \end{cases}$$

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

Implement ANDNOT function using M-P Neuron Model(Use Binary Data)

Implement AND NOT function using M-P Neuron Model(Use Binary Data)

(eg, AND NOT eg.)



Truth table of ANDNOT

x_1	x_2	Not x_2	y
0	0	1	0
0	1	0	0
1	0	1	1
1	1	0	0

From the truth table, it is clear that for ANDNOT function the o/p becomes will fire only if $x_1 = 1$ and $x_2 = 0$.

Implement AND NOT function using M-P Neuron Model(Use Binary Data)

Case 1:- Assume that all the weights are excitatory i.e., $w_1 = w_2 = 1$.
The net i/p calculated for four i/p pairs are:-

$$\begin{aligned} (0,0) &\Rightarrow Y_{in} = x_1 w_1 + x_2 w_2 = 0 \times 1 + 0 \times 1 = 0 \\ (0,1) &\Rightarrow Y_{in} = x_1 w_1 + x_2 w_2 = 0 \times 1 + 1 \times 1 = 1 \\ (1,0) &\Rightarrow Y_{in} = x_1 w_1 + x_2 w_2 = 1 \times 1 + 0 \times 1 = 1 \\ (1,1) &\Rightarrow Y_{in} = x_1 w_1 + x_2 w_2 = 1 \times 1 + 1 \times 1 = 2 \end{aligned}$$

From the calculated net i/p's, it is not possible to differentiate Y_{in} for $x_1 = 1$ and $x_2 = 0$ (for which the neuron fires).

Hence these weights are not suitable.

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

Implement AND NOT function using M-P Neuron Model(Use Binary Data)

Case 2:- Assume one weight as excitatory and the other as inhibitory i.e., $w_1 = 1$, $w_2 = -1$.

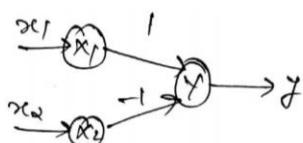
The net i/p calculated for four i/p pairs are:-

$$\begin{aligned} (0,0) &\Rightarrow Y_{in} = x_1 w_1 + x_2 w_2 = 0 \times 1 + 0 \times -1 = 0 \\ (0,1) &\Rightarrow Y_{in} = x_1 w_1 + x_2 w_2 = 0 \times 1 + 1 \times -1 = -1 \\ (1,0) &\Rightarrow Y_{in} = x_1 w_1 + x_2 w_2 = 1 \times 1 + 0 \times -1 = 1 \\ (1,1) &\Rightarrow Y_{in} = x_1 w_1 + x_2 w_2 = 1 \times 1 + 1 \times -1 = 0 \end{aligned}$$

From the calculated net inputs, it is possible to differentiate Y_{in} for $x_1 = 1$ and $x_2 = 0$.

Thus if we set $0 \geq 1$ as threshold only Y_{in} with $x_1 = 1$ and $x_2 = 0$ will produce o/p as 1. i.e., $w_1 = 1$, $w_2 = -1$ with $0 \geq 1$

Implement AND NOT function using M-P Neuron Model(Use Binary Data)



Ktunotes.in

and the net o/p, $y = f(Y_{in})$

$$= \begin{cases} 1 & \text{if } Y_{in} \geq 1 \\ 0 & \text{if } Y_{in} < 1 \end{cases}$$

$$\begin{array}{|c|} \hline \theta > nw - p \\ \hline \end{array}$$

$n=2$ $w=1$ $p=1$ $\text{so } nw-p=2 \times 1 - 1 = 1$

Implement NOT function using M-P Neuron Model(with one input)

Implement NOT gate function using M-P Neuron Model (with one input x_1)

Truth table

x_1	y
0	1
1	0



From the truth table, it is clear that for NOT function the op neuron will fire only if $x_1 = 0$

Case 1:- Assume that w_1 is excitatory

$$x_1 \cdot w_1 = 1$$

The net IP calculated for four IP pairs are :-

$$(0) \Rightarrow y_{in} = x_1 w_1 = 0 \times 1 = 0$$

$$(1) \Rightarrow y_{in} = x_1 w_1 = 1 \times 1 = 1$$

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

Implement NOT gate function using M-P Neuron Model (with one input x_1)

From the calculated net IP's, it is not possible to set a threshold value for $x_1 = 0$. (for which the neuron fires). Hence these weights are not suitable.

Case 2:- Assume that w_1 is inhibitory i.e., $w_1 = -1$

The net IP calculated for four IP pairs are :-

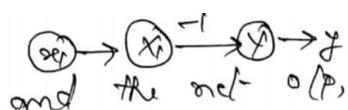
$$(0) \Rightarrow y_{in} = x_1 w_1 = 0 \times -1 = 0$$

$$(1) \Rightarrow y_{in} = x_1 w_1 = 1 \times -1 = -1$$

From the calculated net IP's, it is possible to set y_{in}

and for $x_1 = 0$ thus the threshold value, $0 \geq 0$, i.e. $w_1 = -1$ with $0 \geq 0$

Implement NOT gate function using M-P Neuron Model (with one input x_1)



and the net IP,

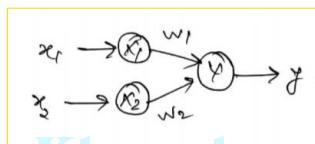
$$y = f(y_{in})$$

$$= \begin{cases} 1 & \text{if } y_{in} \geq 0 \\ 0 & \text{if } y_{in} < 0 \end{cases}$$

Implement OR function using M-P Neuron Model(Use Binary Data)

Implement OR gate function using M-P Neuron Model(with binary data)

Truth table		
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1



Ktunotes.in

From the truth table, it is clear that for OR function the OP neurons will not fire if $x_1 = 0$ and $x_2 = 0$ and will fire for the remaining cases.

Implement OR gate function using M-P Neuron Model(with binary data)

case:- Assume that all the weights are excited by, i.e., $w_1 = w_2 = 1$.
The net ip calculated for four input pairs

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

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

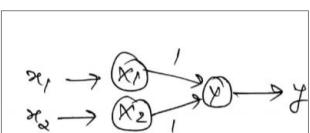
$$(1,0) \Rightarrow y_{in} = x_1 w_1 + x_2 w_2 = 1 \times 1 + 0 \times 1 = 1$$

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

From, the y_{in} calculated it is clear that if we set threshold $\theta \geq 1$ then the truth table is satisfied. i.e., The neuron will fire if $y_{in} \geq 1$, otherwise will not fire. i.e., $w_1 = w_2 = 1$ with $\theta \geq 1$

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

Implement OR gate function using M-P Neuron Model(with binary data)



$$kw \geq \theta > (k-1)w$$

$$\begin{aligned} kw &= 2 \times 1 = 2 \\ (k-1)w &= (2-1) \times 1 = 1 \end{aligned}$$

Ktunotes.in

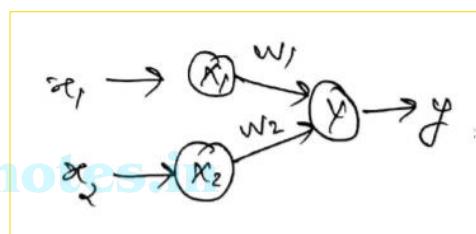
$$\theta > nw - p$$

$$\begin{aligned} \text{the net op, } y &= f(y_{in}) \\ &= \begin{cases} 1 & \text{if } y_{in} \geq 1 \\ 0 & \text{if } y_{in} < 1 \end{cases} \end{aligned}$$

Implement NAND function using M-P Neuron Model(Use Binary Data)

Implement NAND(NOT-AND) gate function using M-P Neuron Model(with binary input)

Truth table		
x_1	x_2	y
0	0	1
0	1	1
1	0	1
1	1	0



From the truth table, it is clear that for NAND function the o/p neuron will fire only if $x_1=1$ and $x_2=1$ and will fire for the remaining cases.

Implement NAND(NOT-AND) gate function using M-P Neuron Model(with binary input)

Case 1:- Assume that all the weights are excitatory i.e., $w_1 = w_2 = 1$. The net i/p calculated for the four i/p pairs are:-

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

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

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

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

From the calculated net i/p's, it is not possible to set a threshold value for $(x_1, x_2) = (0,0), (0,1)$ and $(1,0)$

Truth table			y_{in}
x_1	x_2	y	y_{in}
0	0	1	0
0	1	1	1
1	0	1	1
1	1	0	2

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

Implement NAND(NOT-AND) gate function using M-P Neuron Model(with binary input)

Case 2:- Assume that the weights are $w_1 = 1$ and $w_2 = -1$

The net i/p calculated for four i/p pairs are

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

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

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

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

Truth table			y_{in}
x_1	x_2	y	y_{in}
0	0	1	0
0	1	1	-1
1	0	1	1
1	1	0	0

From the calculated net i/p's, it is not possible to set a threshold value

Implement NAND(NOT-AND) gate function using M-P Neuron Model(with binary input)

Case 3:- Assume that $w_1 = -1$ and $w_2 = 1$.

The net i/p calculated for four i/p pairs are

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

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

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

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

Truth table			y_{in}
x_1	x_2	y	y_{in}
0	0	1	0
0	1	1	1
1	0	1	-1
1	1	0	0

From the calculated net i/p's, it is not possible to set a threshold value

Implement NAND(NOT-AND) gate function using M-P Neuron Model(with binary input)

Case 4:-

Assume that both the weights are inhibitory i.e., $w_1 = w_2 = -1$
The net o/p calculated for four i/p pairs are:-

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

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

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

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

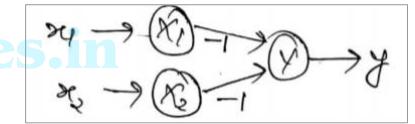
Truth table			Yin
\bar{x}_1	\bar{x}_2	y	
0	0	1	0
0	1	1	-1
1	0	1	-1
1	1	0	-2

Implement NAND(NOT-AND) gate function using M-P Neuron Model (with binary input)

From the calculated net o/p's, we can set the threshold value, θ as $0 \geq -1$ with $w_1 = w_2 = -1$

The net o/p calculated as

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



$$\begin{aligned} \theta &> nw - p \\ &= 2 \cdot 0 - 1 \\ &= -1 \end{aligned}$$

Implement XOR function using M-P Neuron Model(Use Binary Data)

Implement XOR gate function using M-P Neuron Model (with binary input)

Solution:

The truth table for XOR function is given

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

In this case, the output is "ON" for only odd number of 1's.

For the rest it is "OFF."

XOR function cannot be represented by simple and single logic function;
it is represented as

$$y = x_1 \bar{x}_2 + \bar{x}_1 x_2$$

$$y = z_1 + z_2$$

where

$$z_1 = x_1 \bar{x}_2 \quad (\text{function 1})$$

$$z_2 = \bar{x}_1 x_2 \quad (\text{function 2})$$

$$y = z_1 (\text{OR}) z_2 \quad (\text{function 3})$$

Implement XOR gate function using M-P Neuron Model (with binary input)

First function ($z_1 = x_1 \bar{x}_2$):

The truth table for function z_1

x_1	x_2	z_1
0	0	0
0	1	0
1	0	1
1	1	0

The net representation is given as

Case 1: Assume both weights as excitatory, i.e.,

$$w_{11} = w_{21} = 1$$

Calculate the net inputs. For inputs,

$$\begin{aligned}(0,0), z_{1in} &= 0 \times 1 + 0 \times 1 = 0 \\ (0,1), z_{1in} &= 0 \times 1 + 1 \times 1 = 1 \\ (1,0), z_{1in} &= 1 \times 1 + 0 \times 1 = 1 \\ (1,1), z_{1in} &= 1 \times 1 + 1 \times 1 = 2\end{aligned}$$

Hence, it is not possible to obtain function z_1 using these weights.

Ktunotes.in

Implement XOR gate function using M-P Neuron Model (with binary input)

First function ($z_1 = x_1 \bar{x}_2$):

The truth table for function z_1

x_1	x_2	z_1
0	0	0
0	1	0
1	0	1
1	1	0

The net representation is given as

Case 2: Assume one weight as excitatory and the other as inhibitory, i.e.,

$$w_{11} = 1; w_{21} = -1$$

Calculate the net inputs. For inputs

$$\begin{aligned}(0,0), z_{1in} &= 0 \times 1 + 0 \times -1 = 0 \\ (0,1), z_{1in} &= 0 \times 1 + 1 \times -1 = -1 \\ (1,0), z_{1in} &= 1 \times 1 + 0 \times -1 = 1 \\ (1,1), z_{1in} &= 1 \times 1 + 1 \times -1 = 0\end{aligned}$$

On the basis of this calculated net input, it is possible to get the required output.

Hence, $w_{11} = 1$

$$w_{21} = -1$$

$\theta \geq 1$ for the Z_1 neuron

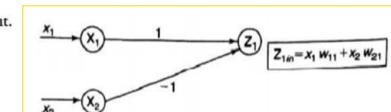


Figure Neural net for Z_1

Implement XOR gate function using M-P Neuron Model (with binary input)

Second function ($z_2 = \bar{x}_1 x_2$):

The net representation is given as follows:
Case 1: Assume both weights as excitatory, i.e.,

The truth table for function z_2

x_1	x_2	z_2
0	0	0
0	1	1
1	0	0
1	1	0

$$w_{12} = w_{22} = 1$$

calculate the net inputs. For the inputs

$$\begin{aligned}(0,0), z_{2in} &= 0 \times 1 + 0 \times 1 = 0 \\ (0,1), z_{2in} &= 0 \times 1 + 1 \times 1 = 1 \\ (1,0), z_{2in} &= 1 \times 1 + 0 \times 1 = 1 \\ (1,1), z_{2in} &= 1 \times 1 + 1 \times 1 = 2\end{aligned}$$

Hence, it is not possible to obtain function z_2 using these weights.

Ktunotes.in

Implement XOR gate function using M-P Neuron Model (with binary input)

Case 2: Assume one weight as excitatory and the other as inhibitory, i.e.,

$$w_{12} = -1; w_{22} = 1$$

calculate the net inputs. For the inputs

$$\begin{aligned}(0,0), z_{2in} &= 0 \times -1 + 0 \times 1 = 0 \\ (0,1), z_{2in} &= 0 \times -1 + 1 \times 1 = 1 \\ (1,0), z_{2in} &= 1 \times -1 + 0 \times 1 = -1 \\ (1,1), z_{2in} &= 1 \times -1 + 1 \times 1 = 0\end{aligned}$$

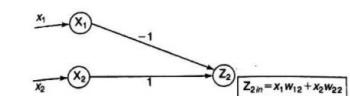


Figure Neural net for Z_2

Thus, based on this calculated net input, it is possible to get the required output, i.e.,

$$w_{12} = -1$$

$$w_{22} = 1$$

$\theta \geq 1$ for the Z_2 neuron

Implement XOR gate function using M-P Neuron Model (with binary input)

Third function ($y = z_1 \text{ OR } z_2$):

The truth table for this function

x_1	x_2	y	z_1	z_2
0	0	0	0	0
0	1	1	0	1
1	0	1	1	0
1	1	0	0	0

Here the net input is calculated using

$$y_{in} = z_1 v_1 + z_2 v_2$$

Case 1: Assume both weights as excitatory, i.e.,

$$v_1 = v_2 = 1$$

Now calculate the net input. For inputs

$$(0, 0), y_{in} = 0 \times 1 + 0 \times 1 = 0$$

$$(0, 1), y_{in} = 0 \times 1 + 1 \times 1 = 1$$

$$(1, 0), y_{in} = 1 \times 1 + 0 \times 1 = 1$$

$$(1, 1), y_{in} = 0 \times 1 + 0 \times 1 = 0$$

(because for $x_1 = 1$ and $x_2 = 1, z_1 = 0$ and $z_2 = 0$)

Implement XOR gate function using M-P Neuron Model (with binary input)

Setting a threshold of $\theta \geq 1, v_1 = v_2 = 1$, which implies that the net is recognized. Therefore, the analysis is made for XOR function using M-P neurons.

Thus for XOR function, the weights are obtained as

$$w_{11} = w_{22} = 1 \text{ (excitatory)}$$

$$w_{12} = w_{21} = -1 \text{ (inhibitory)}$$

$$v_1 = v_2 = 1 \text{ (excitatory)}$$

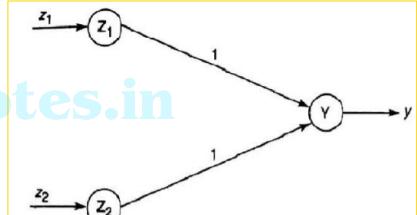
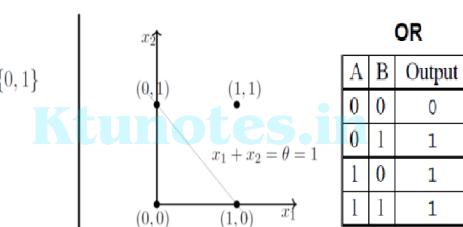
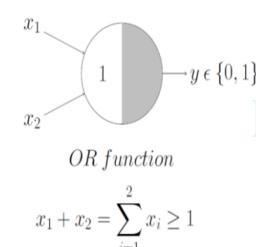


Figure Neural net for $Y(Z_1 \text{ OR } Z_2)$.

McCulloch and Pitts Neuron Conclusion

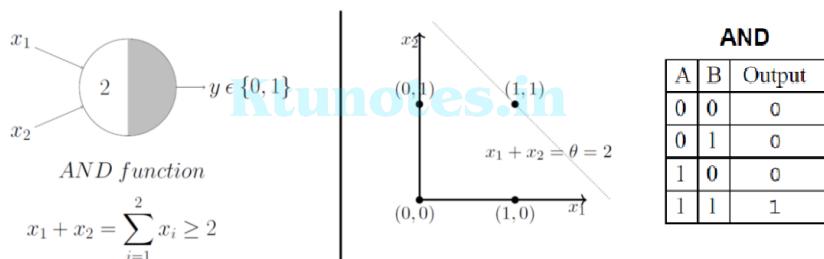
- McCulloch-pitts model doesn't have the ability of learning,because of fixed weight.
- Cannot handle non-boolean (say, real) inputs
- Need to hand code the threshold always
- inputs can either be *excitatory* or *inhibitory*.
- Inhibitory inputs are those that have maximum effect on the decision making irrespective of other inputs
- Excitatory inputs are NOT the ones that will make the neuron fire on their own but they might fire it when combined together

Geometric Interpretation Of M-P Neuron



Geometric Interpretation Of M-P Neuron

AND Function



Hebb Network
Rtunotes.in

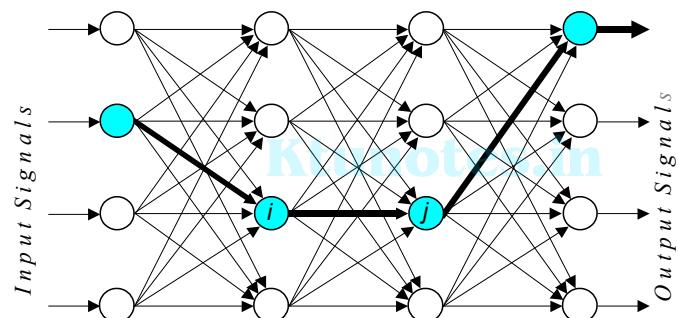
Hebb Network

- In 1949, Donald Hebb proposed one of the key ideas in biological learning, commonly known as Hebb's Law.
- Hebb introduced a neurophysiological **postulate** :
“...When an axon of cell A is near enough to excite a cell B and repeatedly and persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells, such that A's efficiency as one of the cells firing B, is increased.”
- Hebb's Law states that if neuron i is near enough to excite neuron j and repeatedly participates in its activation, **the synaptic connection between these two neurons is strengthened** and neuron j becomes more sensitive to stimuli from neuron i

Hebb Network

- basically the above explanation is derived from the modus operandi (the way in which something operates or works) used by the brain where learning is performed by the changes in the synaptic gap
- In this, if 2 interconnected neurons are ON simultaneously then the weight associated with these neurons can be increased by the modification made in their synaptic gaps(strength).
- Hebb's Law can be represented in the form of two rules:
 - If two neurons on either side of a connection are activated synchronously, then the **weight of that connection is increased**.
 - If two neurons on either side of a connection are activated asynchronously, then the **weight of that connection is decreased**.
- Hebb's Law provides the basis for learning **without a teacher**.
- Learning here is a local phenomenon occurring **without feedback** from the environment.
- It is a kind of **feed-forward, unsupervised learning**.

Hebbian learning in a neural network



Hebb Network

- ▶ Hebb learning rule, is the oldest and simplest ANN rules, was introduced by Donald Hebb in his book *The Organization of Behavior* in 1949.
- ▶ It is more suited for **bipolar data than binary data**
- ▶ From the postulate , we may conclude that the connections between two neurons might be strengthened if the neurons fire at the same time and might weaken if they fire at different times.
- ▶ Hebb Rule states that "the weight vector is found to increase proportionately to the product of the input signal and the learning signal(o/p signal)"
- ▶ **ANN learning rule defines how to adjust the weights of connections to get desirable output**
- ▶ The process of adjusting the weight is known as **Learning**.
- ▶ The procedure to incrementally update each of weights in neural is based on **Training algorithm & Learning law**

Hebb Network

In Hebb learning, if the two interconnected neurons are *on* simultaneously, then the weight associated with these neurons can be increased by the modification made in their weight.

The weight update is given by,

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

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

Hebb Network

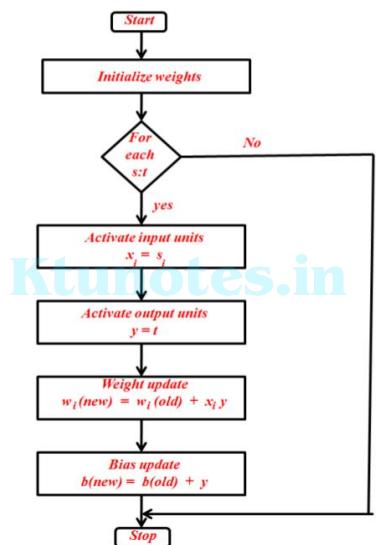
In Hebb learning, if the two interconnected neurons are *on* simultaneously, then the weight associated with these neurons can be increased by the modification made in their weight.

The weight update is given by,

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

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

Hebb Network Flowchart& Algorithm



Hebb Network-Training Algorithm

Step 0: Initialize the weights.

$$w_i = 0 \text{ for } i = 1 \text{ to } n$$

Step 1: Steps 2–4 have to be performed for each input training vector and target output pair, $s : t$.

Step 2: Input units activations are set.

$$x_i = s_i \text{ for } i = 1 \text{ to } n$$

Step 3: Output units activations are set. $y = t$

Step 4: Weight adjustments and bias adjustments are performed.

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

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

$$\text{Change in weight, } \Delta w = xy \quad \Delta b = y$$

Hebb Network Problem

Hebb Network -logical AND function

Design a Hebb Network to implement logical AND function .Use bipolar i/p and targets

2 Input AND Gate
TRUTH TABLE

INPUTS		OUTPUT
X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

Training data table			
Inputs			Target
x1	x2	b	y
1	1	1	1
1	-1	1	-1
-1	1	1	-1
-1	-1	1	-1

► AND function is very simple and mostly known to everyone where the output is **1/SET/ON** if both the inputs are **1/SET/ON**.

But in this example, we have used **-1** instead of '0' this is because the Hebb network uses bipolar data and not binary data because the product item in the above equations would give the output as **0** which leads to a wrong calculation.

Design a Hebb Network to implement logical AND function .Use bipolar i/p and targets

► Step 1: Initially the weight and bias as set as Zero. $w_1=w_2=b=0$

► Step2 :First input $[x_1 \ x_2 \ b]=[1 \ 1 \ 1]$ target $y=1$.

Setting the initial weight as old weight and applying the Hebb Rule.

$$((w_1(\text{old})=0, w_2(\text{old})=0, b(\text{old})=0))$$

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

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

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

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

Training data table			
Inputs			Target
x1	x2	b	y
1	1	1	1
1	-1	1	-1
-1	1	1	-1
-1	-1	1	-1

Design a Hebb Network to implement logical AND function .Use bipolar i/p and targets

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

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

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

► The above calculated weights are used as the initial weight when second pattern is considered.

Inputs			Weight changes		Weights				
x1	x2	b	y	Δw_1	Δw_2	Δb	w_1	w_2	b
1	1	1	1	1	1	1	1	1	1
1	-1	1	-1	1	1	1	1	1	1
-1	1	1	-1	-1	1	1	1	1	1
-1	-1	1	-1	-1	1	1	1	1	1

Design a Hebb Network to implement logical AND function .Use bipolar i/p and targets

- Second input $[x_1 \ x_2 \ b] = [1 \ -1 \ 1]$ target $y = -1$.

Setting the initial weight as old weight and applying the Hebb Rule
 $((w_1(\text{old})=1, w_2(\text{old})=1, b(\text{old})=1))$

- The weight changes here are:

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

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

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

$$\Delta b = y = -1$$

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

$$w_2(\text{new}) = w_2(\text{old}) + \Delta w_2 = 1 + 1 = 2$$

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

Training data table						
Inputs			Weight changes			Target
x_1	x_2	b	y	Δw_1	Δw_2	Δb
1	1	1	1	1	1	1
1	-1	1	-1	-1	1	-1
-1	1	1	-1	1	-1	-1
-1	-1	1	-1	1	1	-1

Inputs							Weight changes			Weights
x_1	x_2	b	y	Δw_1	Δw_2	Δb	w_1	w_2	b	
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	1	1	-2	

Ktunotes.in

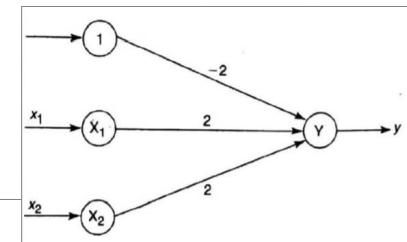
Design a Hebb Network to implement logical AND function .Use bipolar i/p and targets

Inputs		Weight changes			Weights				
x_1	x_2	b	y	Δw_1	Δw_2	Δb	w_1	w_2	b
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	1	1	-2

Inputs		Weight changes			Weights				
x_1	x_2	b	y	Δw_1	Δw_2	Δb	w_1	w_2	b
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	1	1	-2

$$w_1 = 2; \quad w_2 = 2; \quad b = -2$$

The network can be represented as shown in Figure



Hebb Network -logical OR function

Design a Hebb Network to implement logical OR function
Use bipolar i/p and targets

Truth Table		
A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

INPUTS		TARGETS
x_1	x_2	t
-1	-1	-1
-1	1	1
1	-1	1
1	1	1

INPUTS		bias	TARGETS
x_1	x_2	b	t
-1	-1	1	-1
-1	1	1	1
1	-1	1	1
1	1	1	1

Step 1: Initially the weight and bias as set as Zero. $w_1=w_2=b=0$

Step2 :First input $[x_1 \ x_2 \ b]=[-1 \ -1 \ 1]$ target $y=-1$.

Design a Hebb Network to implement logical OR function Use bipolar i/p and targets

Setting the initial weight as old weight and applying the Hebb Rule.

$$((w_1(\text{old})=0, w_2(\text{old})=0, b(\text{old})=0))$$

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

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

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

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

The above calculated weights are used as the initial weight when second pattern is considered. Weight change here is $\Delta w_i = x_i y$

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

INPUTS		bias	TARGETS	w1	w2	b
x1	x2	b	t	w1	w2	b
-1	-1	1	-1			
-1	1	1	1			
1	-1	1	1			
1	1	1	1			

Design a Hebb Network to implement logical OR function Use bipolar i/p and targets

The weight changes here are:

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

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

$$\Delta b = y = 1$$

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

$$w_2(\text{new}) = w_2(\text{old}) + \Delta w_2 = 1 + 1 = 2$$

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

INPUTS		bias	TARGETS	weight	b _{new}	
x1	x2	b	t	w1	w2	b
-1	-1	1	-1	1	1	-1
-1	1	1	1	0	2	0
1	-1	1	1			
1	1	1	1			

Design a Hebb Network to implement logical OR function Use bipolar i/p and targets

INPUTS		bias	TARGETS	weight	b _{new}	
x1	x2	b	t	w1	w2	b
-1	-1	1	-1	1	1	-1
-1	1	1	1			
1	-1	1	1			
1	1	1	1			

Second input [x1 x2 b] = [-1 1 1] target y=1.

Setting the initial weight as old weight and applying the Hebb Rule.

$$((w_1(\text{old})=1, w_2(\text{old})=1, b(\text{old})=-1))$$

Design a Hebb Network to implement logical OR function Use bipolar i/p and targets

► For third input

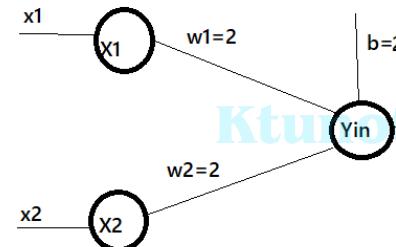
INPUTS		bias	TARGETS	weight	b _{new}	
x1	x2	b	t	w1	w2	b
-1	-1	1	-1	1	1	-1
-1	1	1	1	0	2	0
1	-1	1	1	1	1	1
1	1	1	1			

Design a Hebb Network to implement logical OR function
Use bipolar i/p and targets

► For fourth input

INPUTS		bias	TARGETS	weight		b_{new}
x1	x2	b	t	w1	w2	b
-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	2	2	2

Design a Hebb Network to implement logical OR function
Use bipolar i/p and targets



Hebb Network -logical XOR function

Design a Hebb Network to implement XOR function .Use bipolar i/p and targets

Truth Table		
B	A	Q
0	0	0
0	1	1
1	0	1
1	1	0

INPUTS		TARGETS
x1	x2	t
-1	-1	-1
-1	1	1
1	-1	1
1	1	-1

INPUTS		bias	TARGETS
x1	x2	b	t
-1	-1	1	-1
-1	1	1	1
1	-1	1	1
1	1	1	1

Design a Hebb Network to implement XOR function .Use bipolar i/p and targets

Step 1: Initially the weight and bias as set as Zero. $w_1=w_2=b=0$

Step2 :First input $[x_1 \ x_2 \ b]=[-1 \ -1 \ 1]$ target $y=-1$.

Setting the initial weight as old weight and applying the Hebb Rule.
 $((w_1(\text{old})=0, w_2(\text{old})=0, b(\text{old})=0))$

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

Ktunotes.in

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

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

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

Design a Hebb Network to implement XOR function .Use bipolar i/p and targets

The above calculated weights are used as the initial weight when second pattern is considered. Weight change here is $\Delta w_i = x_i y$

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

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

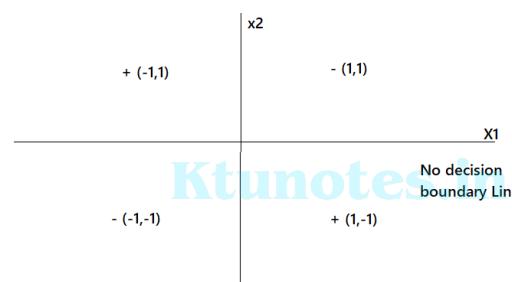
$$\Delta b = y = -1$$

Ktunotes.in

Design a Hebb Network to implement XOR function .Use bipolar i/p and targets

INPUTS		bias	TARGETS	weight		b_{new}
x1	x2	b	t	w1	w2	b
-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	0	0	0

Design a Hebb Network to implement XOR function .Use bipolar i/p and targets



Design a Hebb Network to implement XOR function .Use bipolar i/p and targets

- ▶ The final weights obtained after presenting all i/p functions do not give the correct o/p for all patterns
- ▶ Thus **XOR function is a special case of Pattern classification problem which is not linearly separable.**
- ▶ It can be made linear separable by separating +ve regions and -ve Regions

$$y = x_1 \bar{x}_2 + \bar{x}_1 x_2$$

$$Y = z_1 + z_2$$

$$\text{Where } z_1 = x_1 \bar{x}_2 \text{ & } z_2 = \bar{x}_1 x_2$$

Hebb Network -problems

Using Hebb Network calculate the weight required to perform the following classification of given i/p pattern

+	+	+
+		
+	+	+

+	+	+
+		
+	+	+

I

O

Ktunotes.in

- ▶ The pattern shown is a 3X3 matrix formed in squares.
- ▶ The '+' symbol represent 1 and empty squares represent '-1'.
- ▶ I → belongs to the members of the class → target value 1
- ▶ O → does not belongs to members of class → target value -1

Using Hebb Network calculate the weight required to perform the following classification of given i/p pattern

+	+	+
+		
+	+	+

I

O

Ktunotes.in

Pattern	Inputs										Target
	x1	x2	x3	x4	x5	x6	x7	x8	x9	y	
I	1	1	1	-1	1	-1	1	1	1	1	1
O	1	1	1	1	-1	1	1	1	1	-1	

Using Hebb Network calculate the weight required to perform the following classification of given i/p pattern

- Here a single-layer network with nine input neurons, one bias and one output neuron is formed.
- Set the initial weights and bias to zero, i.e.,
 $w_1=w_2=w_3=w_4=w_5=w_6=w_7=w_8=w_9=0$
- Case 1: Presenting first input pattern (I), we calculate change in weights:

$\Delta w_i = x_i y, \quad i = 1 \text{ to } 9$	$\Delta w_5 = x_5 y = 1 \times 1 = 1$	$\Delta b = y = 1$
$\Delta w_1 = x_1 y = 1 \times 1 = 1$	$\Delta w_6 = x_6 y = -1 \times 1 = -1$	calculate the new weights
$\Delta w_2 = x_2 y = 1 \times 1 = 1$	$\Delta w_7 = x_7 y = 1 \times 1 = 1$	
$\Delta w_3 = x_3 y = 1 \times 1 = 1$	$\Delta w_8 = x_8 y = 1 \times 1 = 1$	
$\Delta w_4 = x_4 y = -1 \times 1 = -1$	$\Delta w_9 = x_9 y = 1 \times 1 = 1$	

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

Using Hebb Network calculate the weight required to perform the following classification of given i/p pattern

Setting the old weights as the initial weights

$$w_1(\text{new}) = w_1(\text{old}) + \Delta w_1 = 0 + 1 = 1$$

$$w_2(\text{new}) = w_2(\text{old}) + \Delta w_2 = 0 + 1 = 1$$

$$w_3(\text{new}) = w_3(\text{old}) + \Delta w_3 = 0 + 1 = 1$$

Similarly, calculating for other weights

$$w_4(\text{new}) = -1, \quad w_5(\text{new}) = 1, \quad w_6(\text{new}) = -1,$$

$$w_7(\text{new}) = 1, \quad w_8(\text{new}) = 1, \quad w_9(\text{new}) = 1, \quad b(\text{new}) = 1$$

weights after presenting first input pattern

weights										Bias
w1	w2	w3	w4	w5	w6	w7	w8	w9	b	
1	1	1	-1	1	1	1	1	1	1	1

$W_{(\text{new})} = [1 \ 1 \ 1 \ -1 \ 1 \ 1 \ 1 \ 1 \ 1]$

Using Hebb Network calculate the weight required to perform the following classification of given i/p pattern

Pattern	Inputs									Target	weights									Bias
	x1	x2	x3	x4	x5	x6	x7	x8	x9		w1	w2	w3	w4	w5	w6	w7	w8	w9	
I	1	1	1	-1	1	-1	1	1	1	1	1	1	1	-1	1	-1	1	1	1	1
O	1	1	1	1	-1	1	1	1	1	-1	1	1	1	1	-1	1	1	1	1	1

Case 2: Now present the second input pattern(O).

The initial weights used here are the final weights obtained after presenting the first input pattern.

the weights are calculated as shown below ($y = -1$ with the initial weights being $[111-11-1111]$).

$$w_i(\text{new}) = w_i(\text{old}) + \Delta x_i; \quad [\Delta w_i = x_i y] \quad w_3(\text{new}) = w_3(\text{old}) + x_3 y = 1 + 1 \times -1 = 0$$

$$w_1(\text{new}) = w_1(\text{old}) + x_1 y = 1 + 1 \times -1 = 0 \quad w_4(\text{new}) = w_4(\text{old}) + x_4 y = -1 + 1 \times -1 = -2$$

$$w_2(\text{new}) = w_2(\text{old}) + x_2 y = 1 + 1 \times -1 = 0 \quad w_5(\text{new}) = w_5(\text{old}) + x_5 y = 1 + -1 \times -1 = 2$$

Using Hebb Network calculate the weight required to perform the following classification of given i/p pattern

Pattern	Inputs									Target	weights									Bias
	x1	x2	x3	x4	x5	x6	x7	x8	x9		w1	w2	w3	w4	w5	w6	w7	w8	w9	
I	1	1	1	-1	1	-1	1	1	1	1	1	1	1	-1	1	-1	1	1	1	1
O	1	1	1	1	-1	1	1	1	1	-1	1	1	1	1	-1	1	1	1	1	1

$$w_6(\text{new}) = w_6(\text{old}) + x_6 y = -1 + 1 \times -1 = -2$$

$$w_7(\text{new}) = w_7(\text{old}) + x_7 y = 1 + 1 \times -1 = 0$$

$$w_8(\text{new}) = w_8(\text{old}) + x_8 y = 1 + 1 \times -1 = 0$$

$$w_9(\text{new}) = w_9(\text{old}) + x_9 y = 1 + 1 \times -1 = 0$$

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

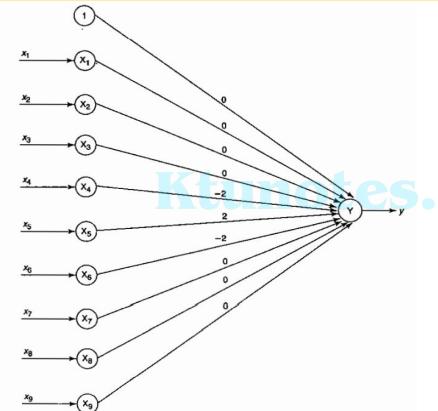
Using Hebb Network calculate the weight required to perform the following classification of given i/p pattern

The final weights after presenting the second input pattern are given as

$$W_{(\text{new})} = [0 \ 0 \ 0 \ -2 \ 2 \ -2 \ 0 \ 0 \ 0]$$

weights										Bias
w1	w2	w3	w4	w5	w6	w7	w8	w9		b
1	1	1	-1	1	-1	1	1	1		1
0	0	0	-2	2	-2	0	0	0		0

Using Hebb Network calculate the weight required to perform the following classification of given i/p pattern



1. Using Hebb Network calculate the weight required to perform the following classification of given i/p pattern

- Using Hebb Network calculate the weight required to perform the following classification of given i/p pattern.
- L → belongs to the members of the class(+) → target value 1
- U → does not belong to members of class(.) → target value -1

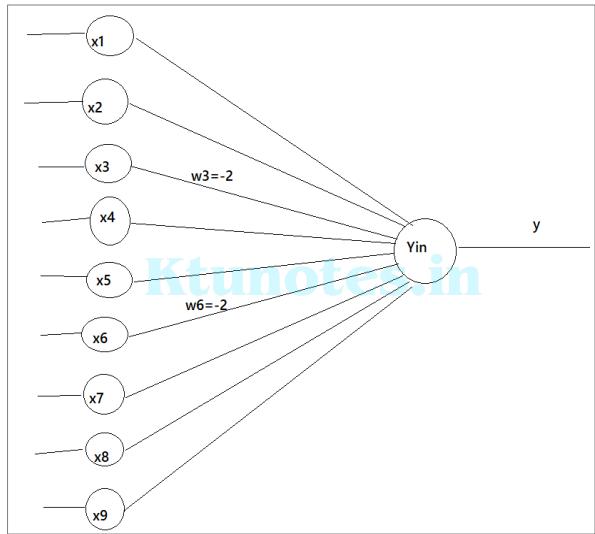
+	.	.
+	.	.
+	+	+

+	.	+
+	.	+
+	+	+

L

U

Hebb Network –tutorial questions
Ktunotes.in



2. Using Hebb Network calculate the weight required to perform the following classification of given i/p pattern

Using Hebb rule, Find the weight required to perform the following classification of the given input patterns shown below

The pattern is shown as 3X3 matrix form in the squares.

The "+" symbol represents the value 1 and dot indicate -1.

Consider "A" belongs to the member of the class so the target value is 1 and "E" does not belong to the member of the class so the target value is -1

+ + +	+ + +
+ . +	+ . .
+ + +	+ + +
+ . +	+ . .
+ . +	+ + +

"A" "E"

3. Using Hebb Network calculate the weight required to perform the following classification of given i/p pattern

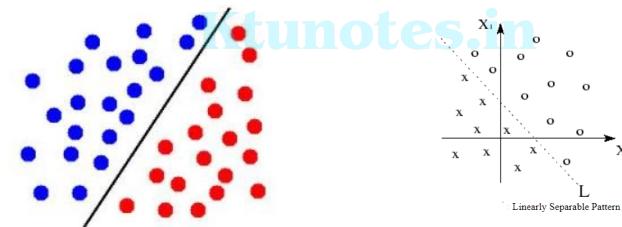
- Using Hebb rule find weights required to perform following classifications.
The vectors (1 -1 1 -1) and (1 1 1 -1) belongs to class (target value +1);
vectors (-1 -1 1 1) and (1 1 -1 -1) do not belongs to class (target value -1).
Also using each of training x vectors as input, test the response of the net

Linear separability

- ▶ ANN does not give exact solution for a nonlinear problem.
- ▶ Linear separability is the concept wherein , the separation of input space into regions is based on whether the network response is positive or negative.
- ▶ Linear separable means ,in the plane if we can splits the input data into two half-spaces such that all points of the first class should be in one half-space and other points of the second class should be in the other half-space.
- ▶ In two dimensional space, it means that there is a line, which separates points of one class from points of the other class.
- ▶ The idea of linearly separable is easiest to visualize and understand in 2 dimensions

Linear separability

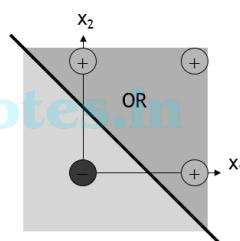
- ▶ A dataset is said to be linearly separable if it is possible to draw a line that can separate the red and green points from each other.
- ▶ For example: In the following image, Let the two classes be represented by colors red and green. if blue circles represent points from one class and red circles represent points from the other class, then these points are linearly separable.



Linear separability

Truth Table		
A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

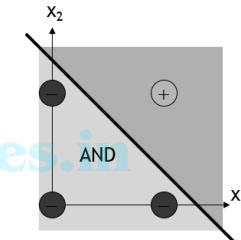
Read as A OR B gives Q



Linear separability

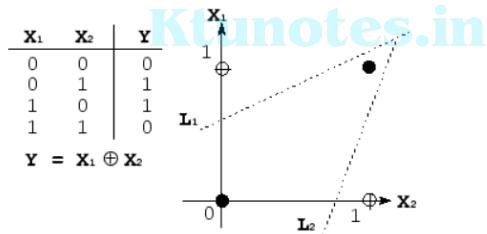
Truth Table		
A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

Read as A AND B gives Q



Linear separability

- The most classic example of linearly inseparable pattern is a logical exclusive-OR (XOR) function.
- The illustration of XOR function that two classes, 0 for black dot and 1 for white dot, cannot be separated with a single line.



Important Questions

- Define Artificial Neural Network
- List the main components of biological neuron.
- Compare and contrast biological neuron and artificial neuron
- State the characteristics of ANN.
- Define Learning. Differentiate between supervised and unsupervised learning
- What is the necessity of Activation function?
- List the commonly used activation functions.
- Define Bias and Threshold.
- Why M-P neuron is widely used in processing binary data?