Module 1- Introduction to Soft Computing and Neural Network

Lecture1 Topic

Introduction to Soft Computing

- Concept of computing
- Important characteristics of "Computing"
- Soft computing vs. "Hard" Computing
- Few examples of Soft computing applications
- Characteristics of Soft computing
- Hybrid computing

Concept of Computing System

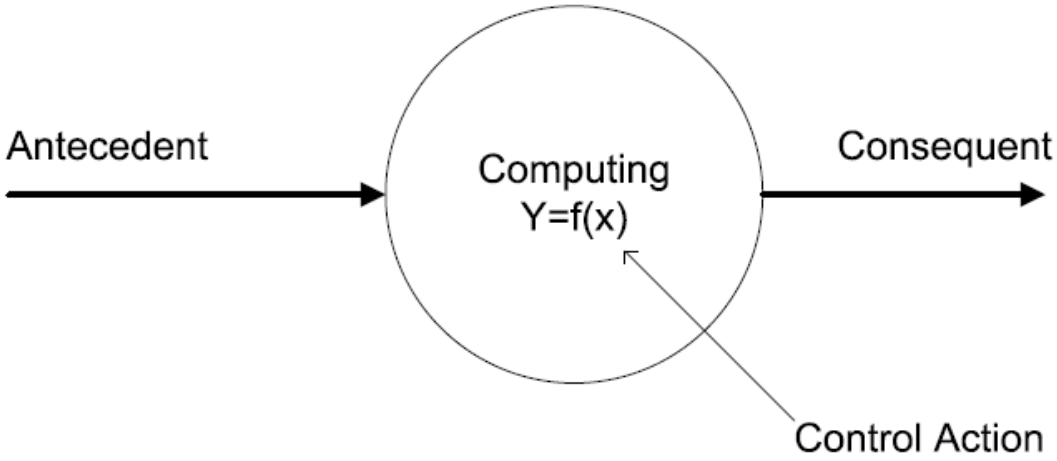


Figure : Basic of computing

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

Important characteristic of computing

- Should provide **precise** solution
- Control action should be **unambiguous** and **accurate**.
- Suitable for problem, which is easy to model **mathematically**.

Hard computing

- In 1996, L.A. Zade (LAZ) introduced the term hard computing.
- According to LAZ: we term a computing as Hard computing,
  - **Precise result** is guaranteed
  - Control action are **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 graph, finding closest pair of points given a set of points, etc.)

Soft Computing

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

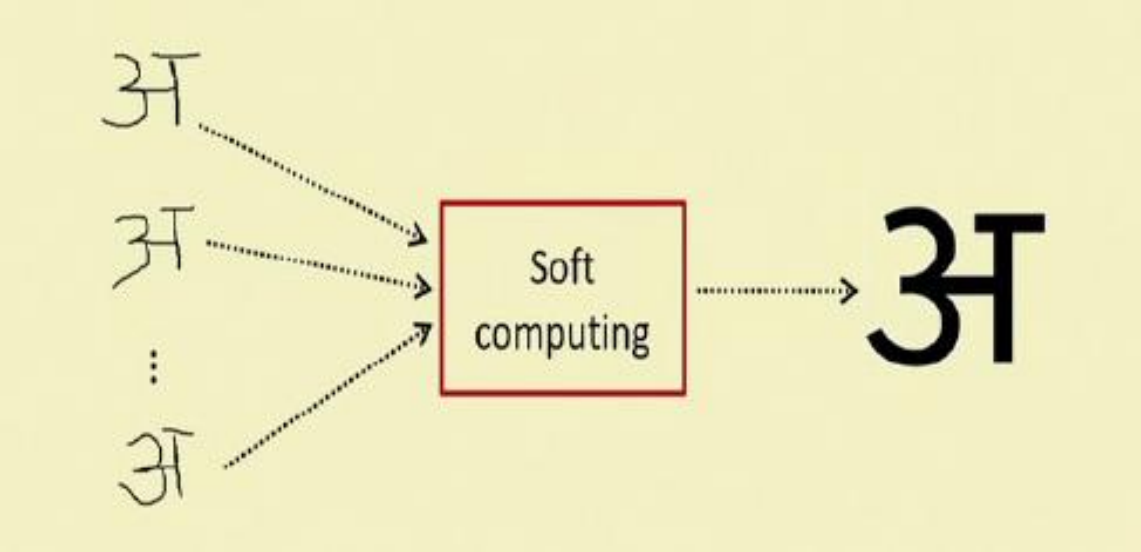
Definition: Soft computing

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. It principal constituents are fuzzy logic, neuro-computing and probablistic reasoning. The role model for soft computing is human mind.

Characteristics of soft computing

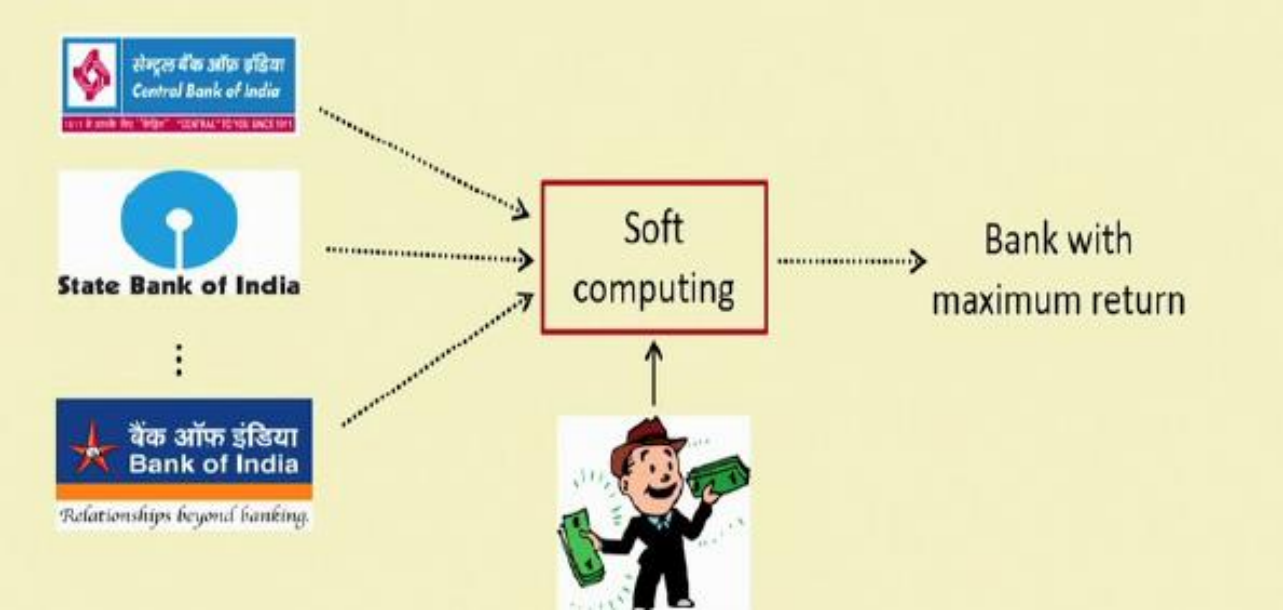
- It **does not require** any mathematical modeling of problem solving.
- It **may not yield** the precise solution.
- Algorithms are **adaptive**(i.e., it can adjust to the change in dynamic environment).
- Use some **biological inspired methodologies** such as genetics, evolution, Ants behaviors, particles swarming, human nervous system etc.)

Examples of soft computing



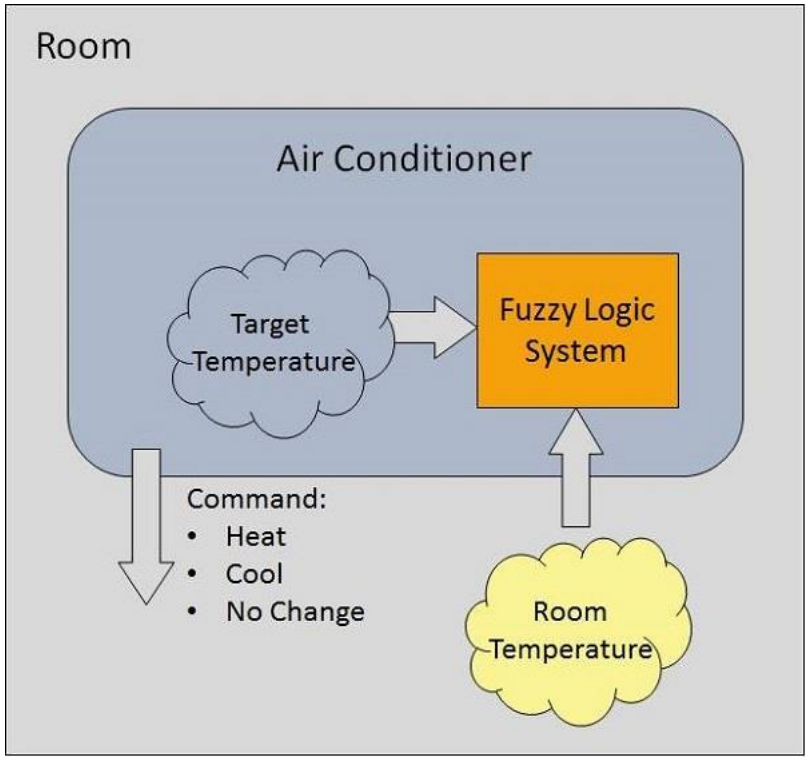
Example: Hand written character recognition
(Artificial Neural networks)

Examples of soft computing



Example: Money allocation problem
(Evolutionary computing)

Examples of soft computing



Example: Air Conditionining System
(Fuzzy logic)

How soft computing?

- How a **student** learns from a **teacher**?
  - Teacher asks questions and tell answer then
  - Teacher puts question and hint answers and asks whether the answers are correct or not.
  - Student thus learns a topic and store in his memory.
  - Based on these knowledge he solves new problems.

- This is way how the human brain works.
- Based on this concept Artificial neural network is used to solve problems

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.
  - Populations is synonymous to solutions.
  - Selection of superior solution is synonymous to exploring the optimal solution.

How soft computing?

- How doctor treats his patient?
  - Doctor asks the patient about suffering.
  - Doctor find the symptoms of diseases
  - Doctor prescribed tests and medicines

- This is exactly how fuzzy logic works.
  - Symptoms are correlated with diseases with uncertainty.
  - Doctors prescribes tests/medicines fuzzily.

Soft Computing

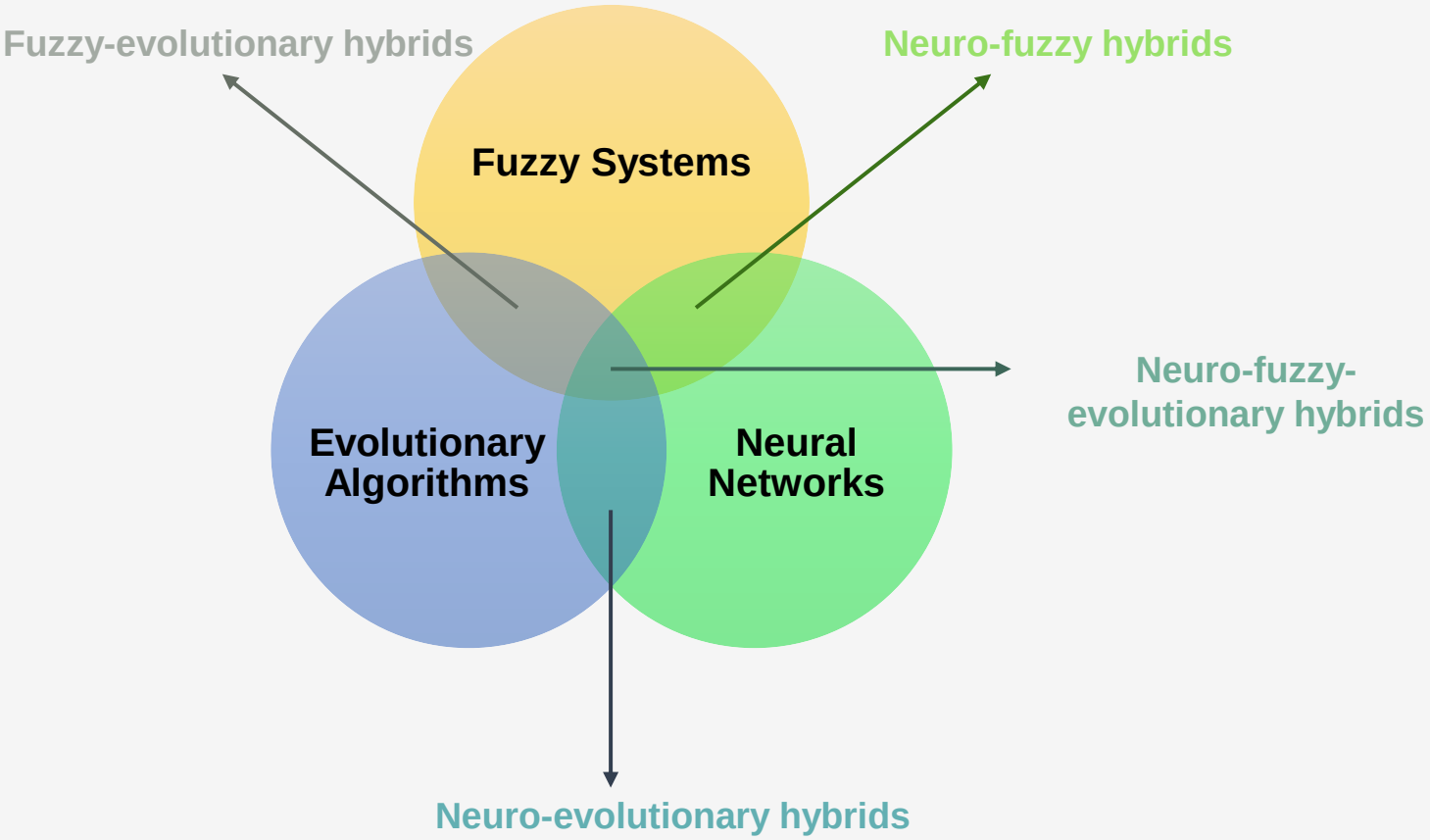
- Introduced by Lotfi A. Zadeh, University of California, Berkley
- Collection of computational methods
- Includes Fuzzy Systems, Neural Networks and Evolutionary Algorithms
- Deployment of *soft computing* for the solution of *machine learning* problems has led to high *Machine Intelligence Quotient*

Image Credit: Electrical Engineering and Computer Sciences, UC. Berkeley

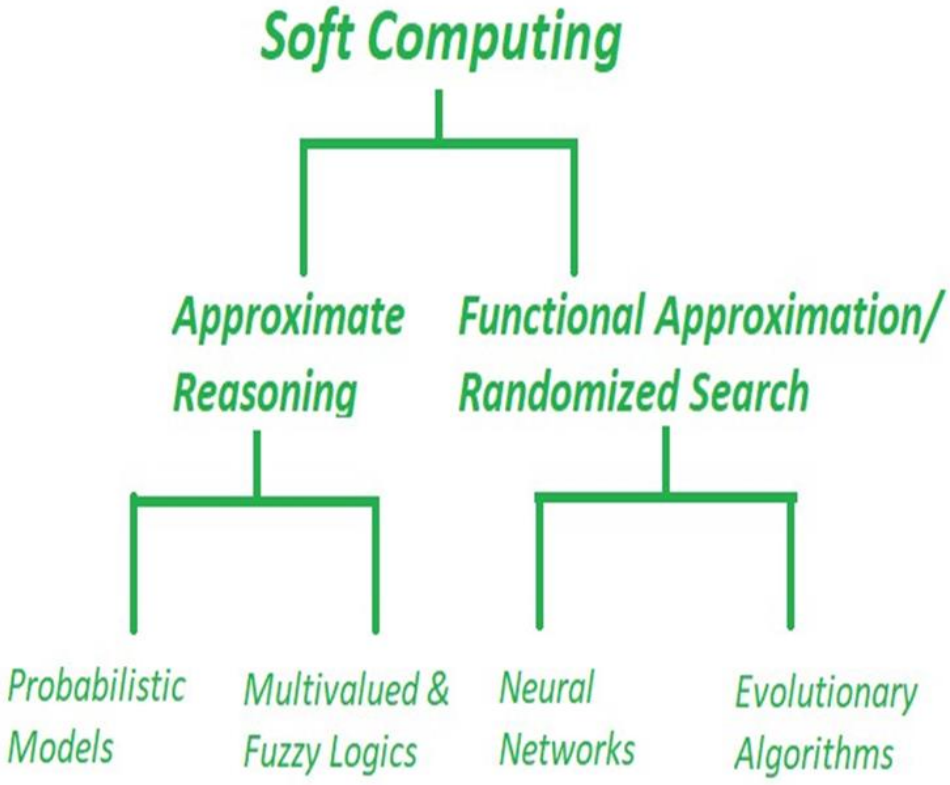
"Soft computing differs from hard computing (conventional computing) in its tolerance to imprecision, uncertainty and partial truth"

-Lotfi A. Zadeh

Soft Computing (Contd…)



Soft computing methodologies

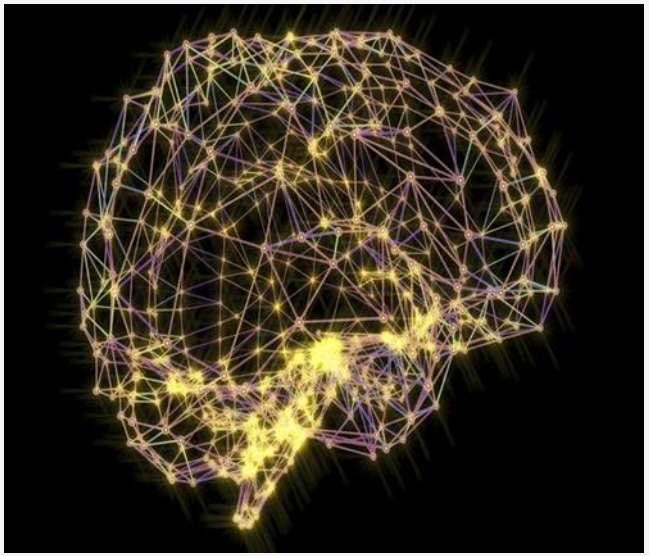


Techniques in soft computing

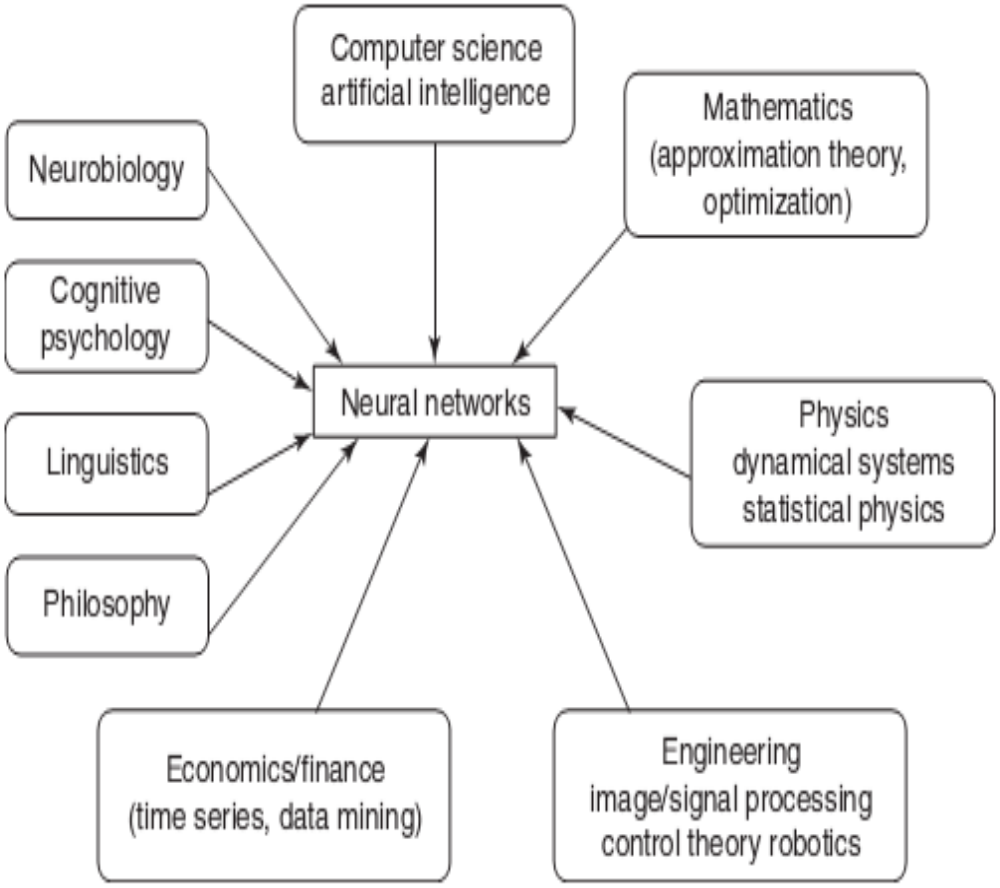
1. Neural Network
2. Fuzzy Logic
3. Genetic Algorithm
4. Hybrid Systems

Neural Networks

- Simplified models of the biological nervous system
- Processing elements called neurons – inspired by the brain
- Parallel distributed processing
- Characteristics:
  - mapping capabilities or pattern association
  - robustness
  - fault tolerance
  - parallel and high speed information processing
  - nonlinearity
  - adaptivity



Multidisciplinary view of neural network



Fuzzy Logic

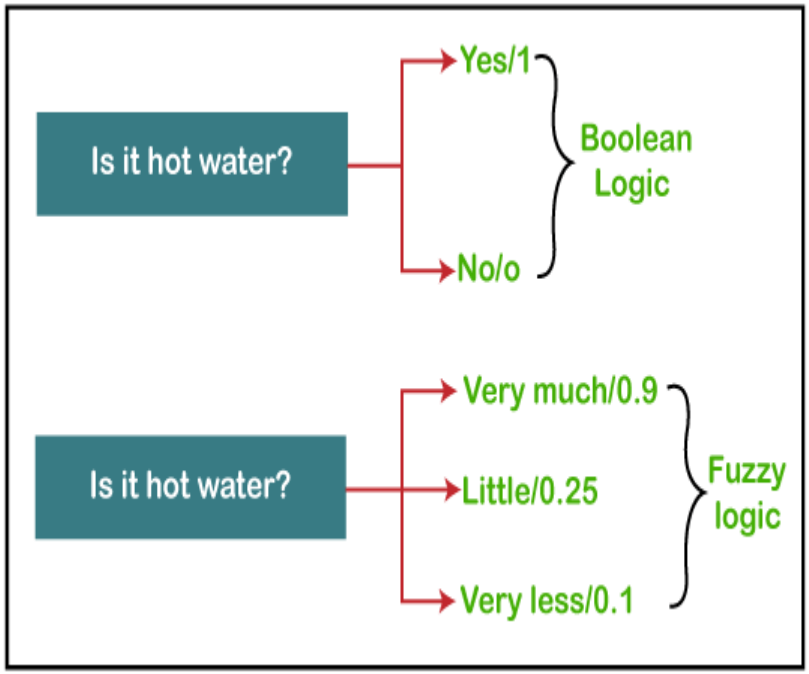
- It was developed in 1965, by Professor Lofti Zadeh, at University of California, Berkley. The first application was to perform computer data processing based on natural values.
- In more simple words, **A Fuzzy logic stat can be 0, 1 or in between these numbers i.e. 0.17 or 0.54**.
- **For example,** In **Boolean**, we may say glass of hot water (i.e 1 or High) or glass of cold water i.e. (0 or low), **but in Fuzzy logic,** We may say glass of warm water (neither hot nor cold).

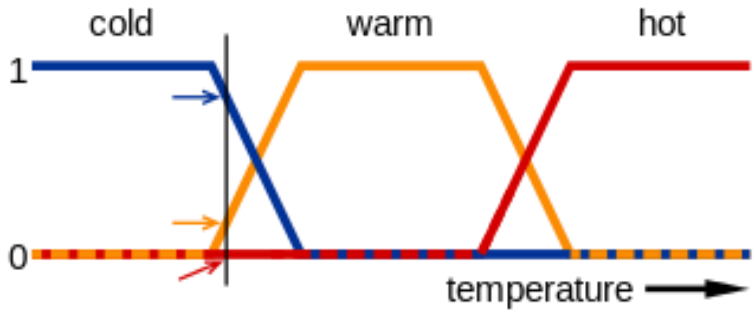
Fuzzy Logic

Fuzzy-"Not Clear, distinct, or precise; blurred"

Fuzzy logic is a reasoning method that is **similar to human *reasoning***. In other words, a fuzzy logic-based system can make decisions similar to a human.

Fuzzy Logic is a technique that understands the vagueness of a solution and presents the solution with a ***degree of vagueness which is practical to human decision***. It is widely applied in several applications of Artificial Intelligence for reasoning.





Advantages of Fuzzy Logic

1. A Fuzzy Logic System is flexible and allow modification in the rules.
2. Even imprecise, distorted and error input information is also accepted by the system.
3. The systems can be easily constructed.
4. Since these systems involve human reasoning and decision making, they are useful in providing solutions to complex solutions in different types of applications.

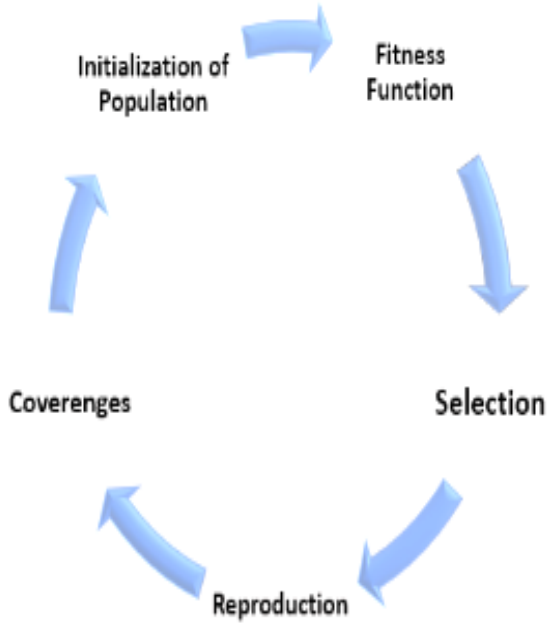
Applications of Fuzzy Logic

- Fuzzy Logic system can be used in Automotive systems, for applications like 4-Wheel steering, automatic gearboxes etc.
- Applications in the field of Domestic Applications include Microwave Ovens, Air Conditioners, Washing Machines, Televisions, Refrigerators, Vacuum Cleaners etc.
- Other applications include Hi-Fi Systems, Photo-Copiers, Humidifiers etc.

Genetic Algorithm

- A genetic algorithm is an adaptive heuristic search algorithm inspired by "Darwin's theory of evolution in Nature."
- In GAs, we have a **pool or a population of possible solutions** to the given problem. These solutions then undergo recombination and mutation (like in natural genetics), producing new children, and the process is repeated over various generations.
- During this procedure a certain strings of symbols, known as chromosomes, evaluate toward better solution.

Genetic Algorithm



- In this way we keep “evolving” better individuals or solutions over generations, till we reach a stopping criterion.

Applications of Genetic Algorithms

- Data mining and clustering.
- Image processing.
- Wireless sensor network.
- Traveling salesman problem (TSP)
- Vehicle routing problems.
- Mechanical engineering design.
- Manufacturing system.

HYBRID SYSTEMS

Hybrid systems enables one to combine various soft computing paradigms and result in a best solution. The major three hybrid systems are as follows:

- Hybrid Fuzzy Logic (FL) Systems
- Hybrid Neural Network (NN) Systems
- Hybrid Evolutionary Algorithm (EA) Systems

Applications of Soft Computing

Hard computing vs Soft computing

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

Hard computing vs Soft computing

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

Hybrid Computing

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

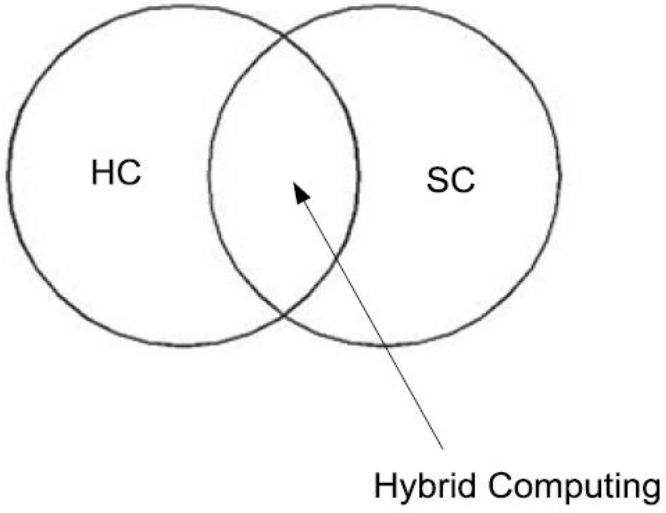


Figure : Concept of Hybrid Computing

Hybrid computing in action

Autonomous Vehicle

- Autonomous vehicles require a combination of various computational techniques to operate effectively:
  - **Neural Networks**: For image and object recognition, helping the vehicle identify pedestrians, other vehicles, and road signs.
  - **Fuzzy Logic**: To make decisions in uncertain environments, such as when to change lanes or how to handle ambiguous traffic situations.
  - **Genetic Algorithms**: To optimize the vehicle's route and improve fuel efficiency.
  - **Traditional Algorithms**: For precise control of vehicle dynamics and ensuring safety through rigorous calculations.

hybrid computing enables autonomous vehicles to navigate complex environments, make real-time decisions, and continuously improve their performance through learning and optimization.

Module 1- Introduction to Soft Computing and Neural Network

Content

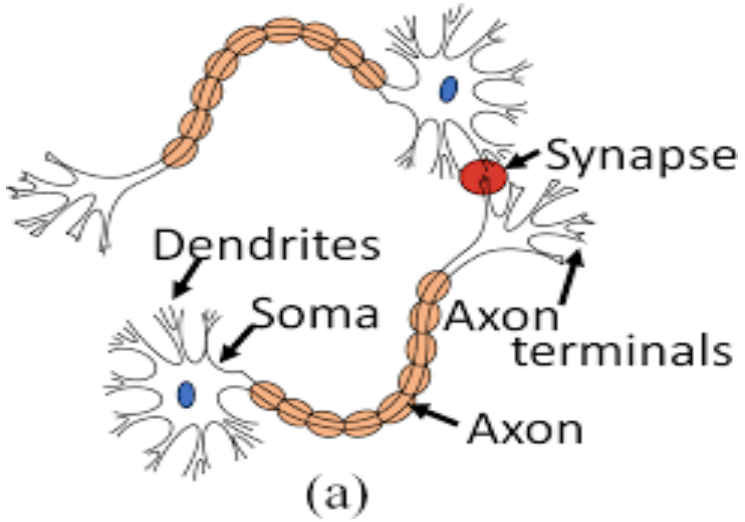
1.2 Biological neurons and its working, ANN – Terminologies, Basic Models (Biological neuron, Aritificial neuron, comprison, ANN, History, NN models(architecture, learning algorithm, activation function), Basic terminolgies

Neural Network

- Biological nervous system **is the most important part** of many living things, in particular, human beings.
- There is a part called **brain** at the center of human nervous system.
- In fact, any biological nervous system consists of a large number of interconnected processing units called **neurons**.
- Each neuron is approximately 10μm long and they can operate in parallel.
- Typically, a human brain consists of approximately 10^{11} neurons communicating with each other with the help of **electrical impulses**.

Neuron:Basic unit of nervous system

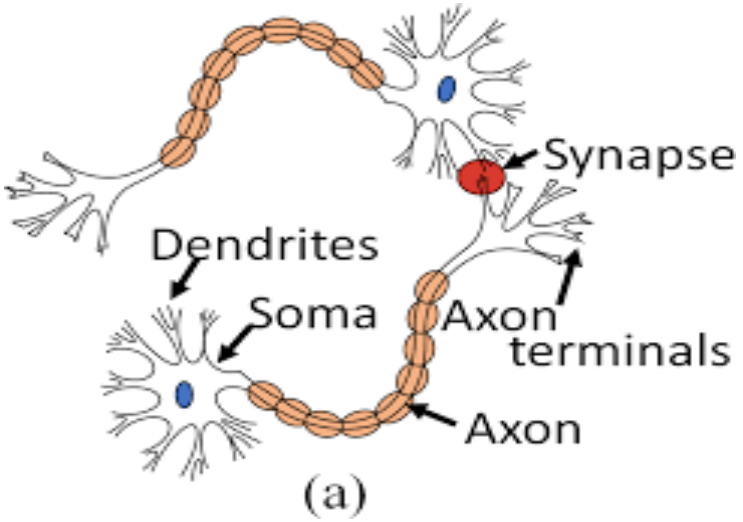
- **Dendrite** : A bush of very thin fibre.
- **Axon** : A long cylindrical fibre.
- **Soma** : It is also called a cell body, and just like as a nucleus of cell.
- **Synapse** : It is a junction where axon makes contact with the dendrites of neighbouring dendrites.



Different parts of a biological neuron

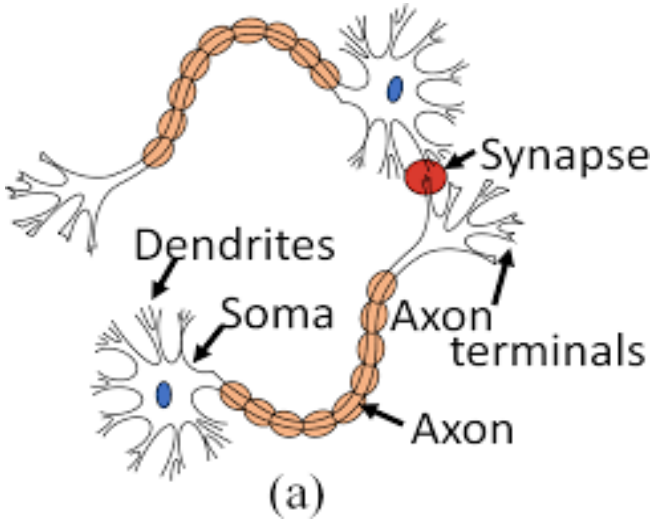
Neuron and its working

- **Dendrite** (Input): receives signals from other neurons
- **Soma** (cell body – processing unit): sums up all signals. It also consists of threshold unit.



Neuron and its working

- **Synapse** (weighted connections): The point of interconnection of one neuron with other neurons. The amount of signal transmitted depends upon the strength (synaptic weight) of the connection. Connection can be Inhibitory (decreasing strength) or Excitatory (increasing strength)
- **Axon** (output): when the sum reaches to the threshold value, neuron fires and generates an output signal.



Biological Neural Network

- *Has three main parts*
  - *Soma or cell body-where cell nucleus is located*
  - *Dendrites-where the nerve is connected to the cell body*
  - *Axon-which carries the impulses of the neuron*
- *Electric impulse is passed between synapse and dendrites.*
- *Synapse- Axon split into strands and strands terminates into small bulb like organs called as synapse.*
- *It is a chemical process which results in increase /decrease in the electric potential inside the body of the receiving cell.*
- *If the electric potential reaches a thresh hold value, receiving cell fires & pulse / action potential of fixed strength and duration is send through the axon to synaptic junction of the cell.*
- *After that, cell has to wait for a period called* ***refractory period.***

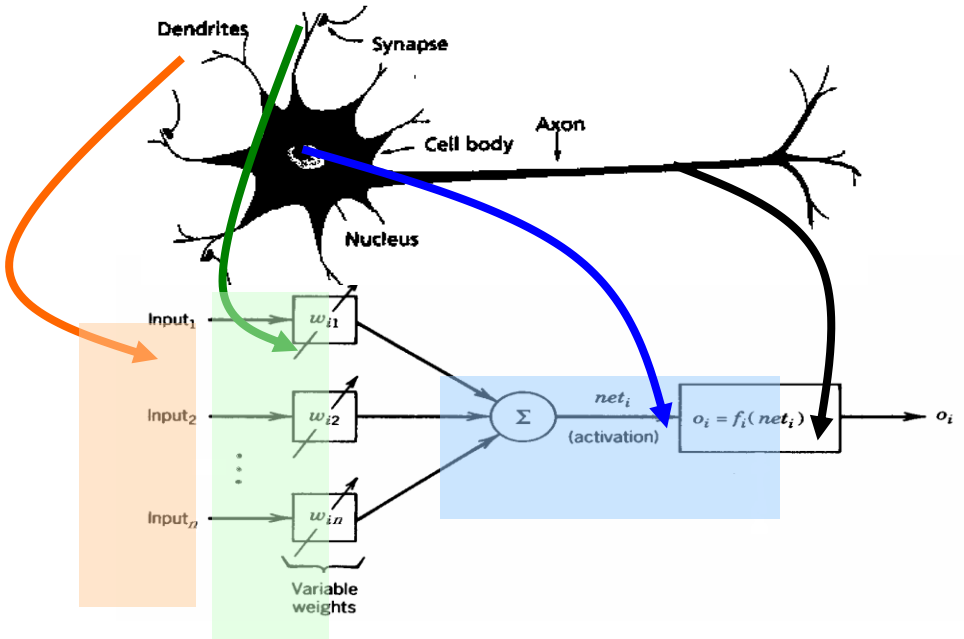
Artificial Neural Network

- ANN is an information processing system that has certain performance characteristics in common with biological nets.
- Several key features of the processing elements of ANN are suggested by the properties of biological neurons:
  - The processing element receives many signals.
  - Signals may be modified by a weight at the receiving synapse.
  - The processing element sums the weighted inputs.
  - Under appropriate circumstances (sufficient input), the neuron transmits a single output.
  - The output from a particular neuron may go to many other neurons.

Artificial Neurons

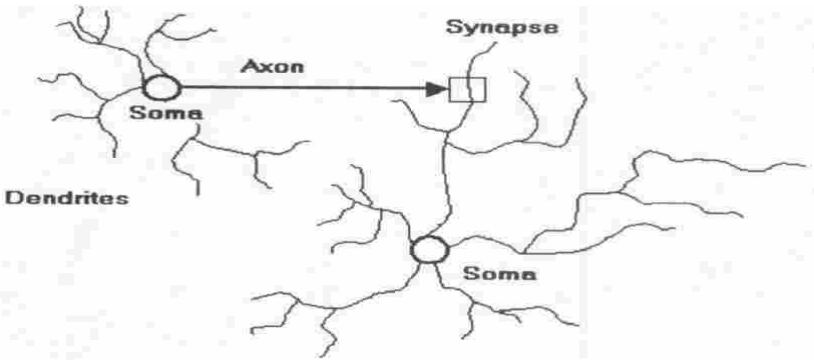
- From experience: examples / training data
- Strength of connection between the neurons is stored as a weight-value for the specific connection.
- Learning the solution to a problem = changing the connection weights

A physical neuron

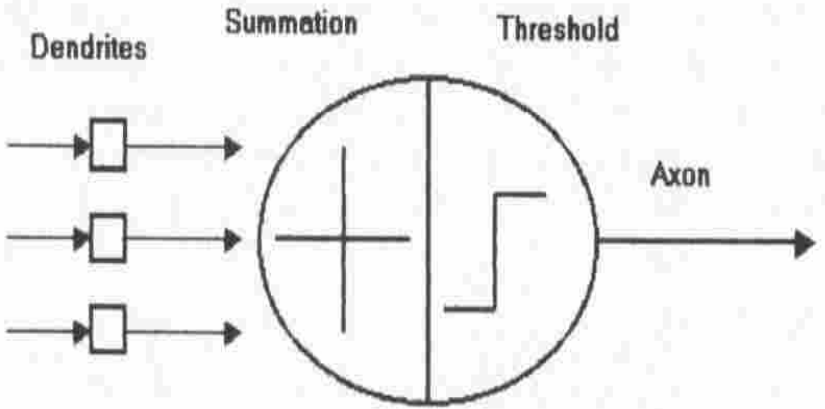


An artificial neuron

Artificial Neurons



Four basic components of a human biological neuron

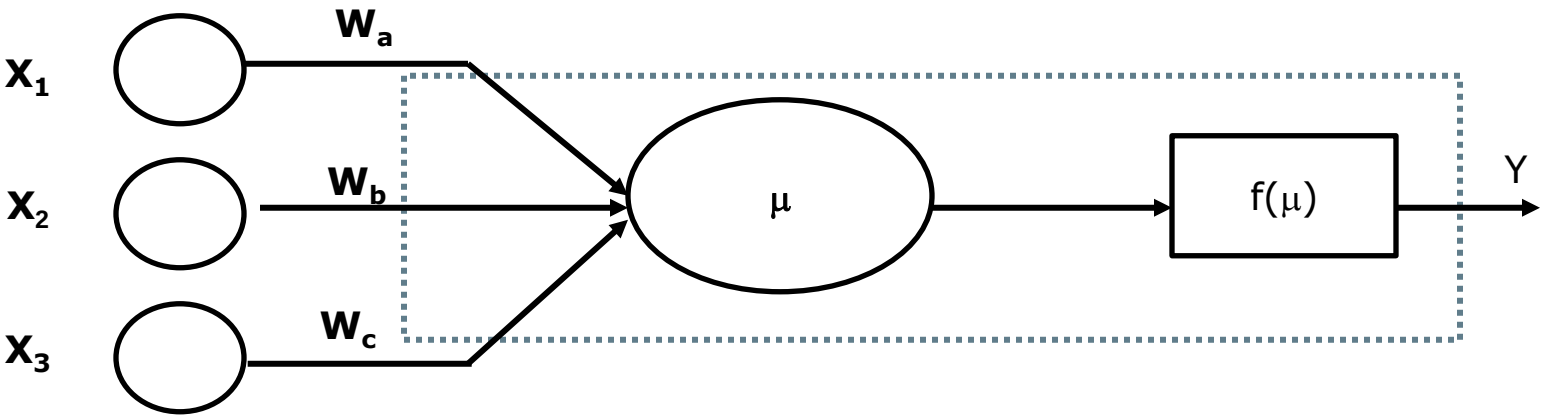


The components of a basic artificial neuron

Terminology Relation Between Biological And Artificial Neuron

Biological Neuron	Artificial Neuron
Cell	Neuron
Dendrites	Weights or interconnections
Soma	Net input
Axon	Output

Model Of A Neuron



Input units

Connection weights

Summing function

computation

(dendrite)

(synapse)

(soma)

(axon)

ANN

- A neural net consists of a large number of simple processing elements called **neurons, units, cells or nodes.**
- Each neuron is connected to other neurons by means of directed communication links, each with **associated weight.**
- The weight represent information being used by the net to solve a problem.
- Each neuron has an internal state, called its **activation or activity level**, which is a function of the inputs it has received. Typically, a neuron sends its activation as a signal to several other neurons.

Brain Vs Computer – Comparison bwn BN and AN

Term	Brain	Computer
Speed	Execution time is few milliseconds	Execution time is few nano seconds
Processing	Perform massive parallel operations simultaneously	Perform several parallel operations simultaneously. It is faster than the biological neuron
Size and complexity	Number of Neuron is 10^{11} and number of interconnections is 10^{15}. So complexity of brain is higher than computer	It depends on the chosen application and network designer.
Storage capacity	i) Information is stored in interconnections or in synapse strength. ii) New information is stored without destroying old one. iii) Sometimes fails to recollect information	i) Stored in continuous memory location. ii) Overloading may destroy older locations. iii) Can be easily retrieved

Contd...

Term	Brain	Computer
Tolerance	i) Fault tolerant ii) Store and retrieve information even interconnections fails iii) Accept redundancies	i) No fault tolerance ii) Information corrupted if the network connections disconnected. iii) No redundancies
Control mechanism	Depends on active chemicals and neuron connections are strong or weak	CPU Control mechanism is very simple

Evolution of neural networks

Year	Neural network	Designer	Description
1943	McCulloch and Pitts neuron	McCulloch and Pitts	Arrangement of neurons is combination of logic gate. Unique feature is thresh hold
1949	Hebb network	Hebb	If two neurons are active, then their connection strengths should be increased.
1958,1959,1962,1988,1960	Perceptron Adaline	Frank Rosenblatt, Block, Minsky and Papert Widrow and Hoff	Weights are adjusted to reduce the difference between the net input to the output unit and the desired output

Contd...

Year	Neural network	Designer	Description
1972	Kohonen self-organizing feature map	Kohonen	Inputs are clustered to obtain a fired output neuron.
1982, 1984, 1985, 1986, 1987	Hopfield network	John Hopfield and Tank	Based on fixed weights. Can act as associative memory nets
1986	Back propagation network	Rumelhart, Hinton and Williams	i) Multilayered ii) Error propagated backward from output to the hidden units

Contd..

1988	Counter propagation network	Grossberg	Similar to kohonen network
1987-1990	Adaptive resonance Theory(ART)	Carpenter and Grossberg	Designed for binary and analog inputs.
1988	Radial basis function network	Broomhead and Lowe	Resemble back propagation network , but activation function used is Gaussian function
1988	Neo cognitron	Fukushima	For character recogniton.

Basic models of ANN

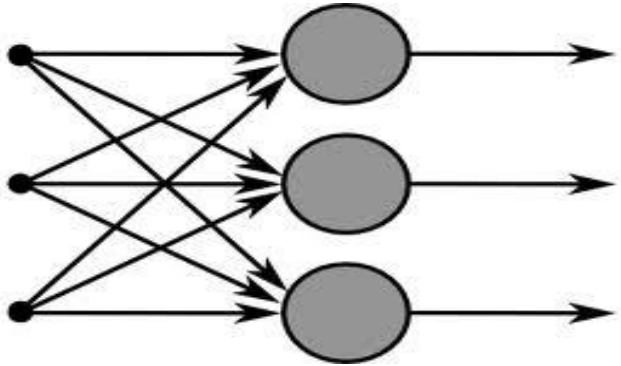
- Models are based on three entities
  - ***The model's synaptic interconnections.***
  - ***The training or learning rules adopted for updating and adjusting the connection weights.***
  - ***Their activation functions***
- The *arrangement of neurons to form layers* and the *connection pattern formed within and between layers* is called the ***network architecture.***

Five types of ANN

1. ***Single layer feed forward network***
2. ***Multilayer feed-forward network***
3. ***Single node with its own feedback***
4. ***Single-layer recurrent network***
5. ***Multilayer recurrent network***

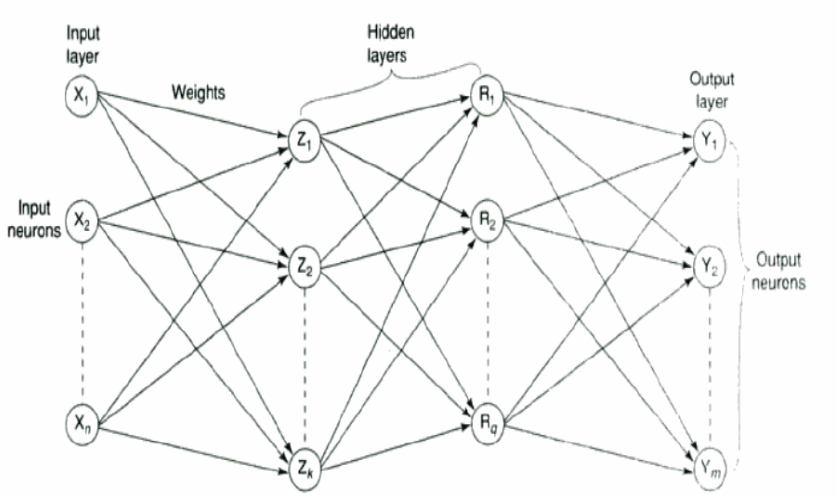
Single layer Feed- Forward Network

- Layer is formed by taking processing elements and combining it with other processing elements.
- Input and output are linked with each other
- Inputs are connected to the processing nodes with various weights, resulting in series of outputs one per node.



Multilayer feed-forward network

- Formed by the interconnection of several layers.
- Input layer receives input and buffers input signal.
- Output layer generates output.
- Layer between input and output is called *hidden layer*.
- Hidden layer is internal to the network.
- Zero to several hidden layers in a network.
- More the hidden layer, more is the complexity of network, but efficient output is produced.



Feed back network

- If no neuron in the output layer is an input to a node in the same layer / proceeding layer – ***feed forward network.***
- If outputs are directed back as input to the processing elements in the same layer/proceeding layer – ***feedback network.***
- If the output are directed back to the input of the same layer then it is ***lateral feedback.***

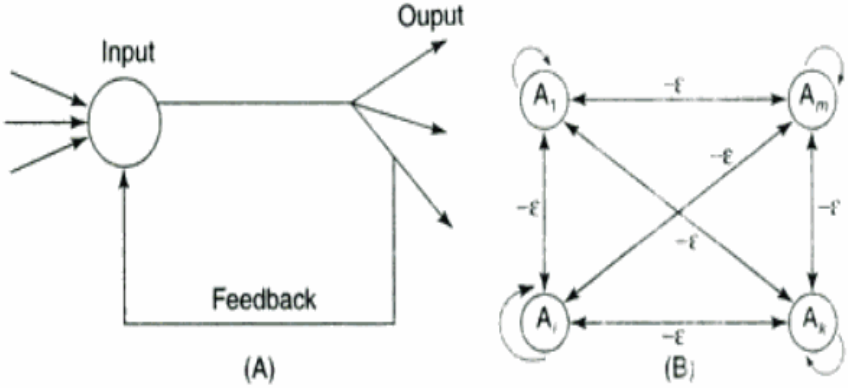


Figure 2-8 (A) Single node with own feedback. (B) Competitive nets.

Recurrent networks are networks with feedback networks with closed loop.
Fig 2.8 (A) –simple recurrent neural network having a single neuron with feedback to itself.
Fig 2.9 – single layer network with feedback from output can be directed to processing element itself or to other processing element/both.

Single layer recurrent layer

Processing element output can be directed to processing element itself or to other processing element in the same layer.

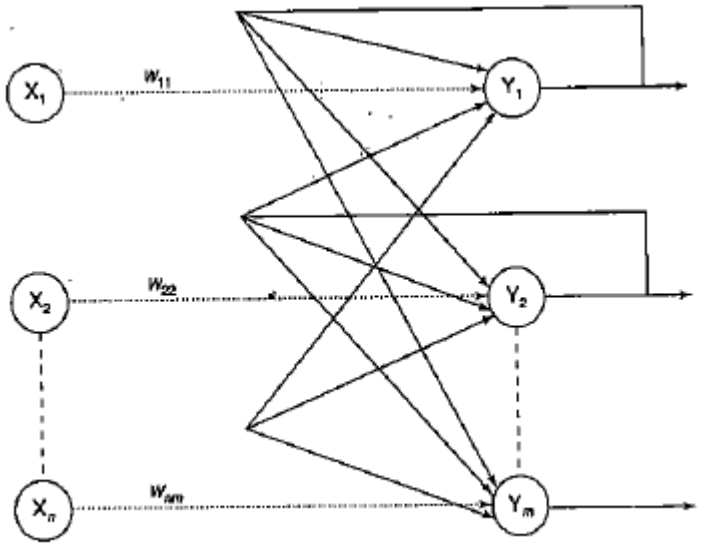


Figure 2-9 Single-layer recurrent network.

Multilayer recurrent network

Processing element output can be directed back to the nodes in the preceding layer, forming a ***multilayer recurrent network.***

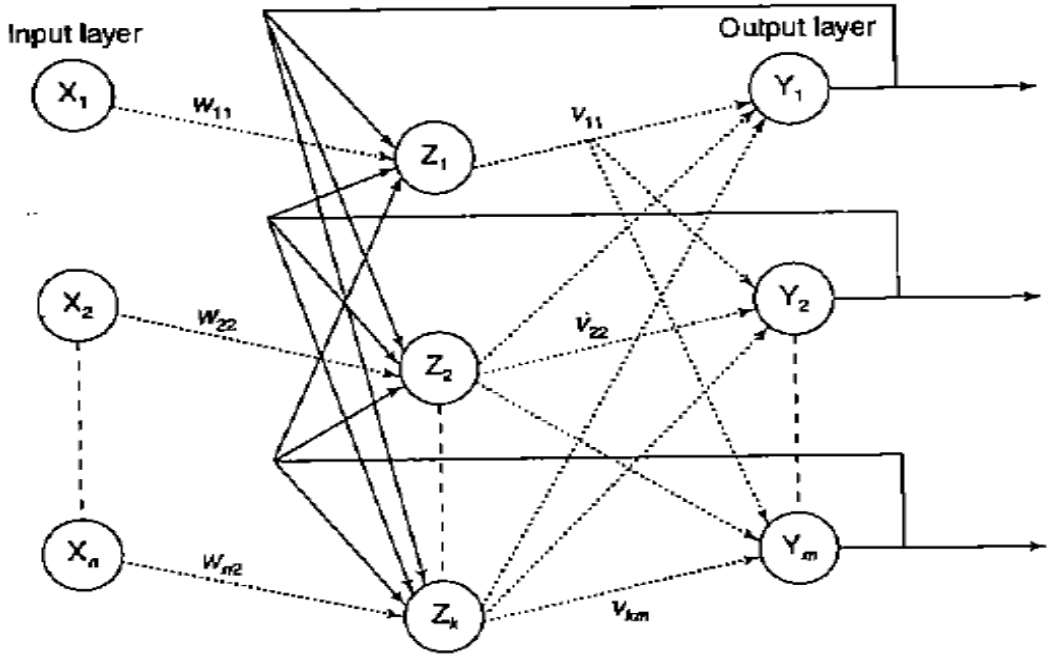


Figure 2-10 Multilayer recurrent network.

Learning

Two broad kinds of learning in ANNs is :

i) parameter learning – updates connecting weights in a neural net.

ii) Structure learning – focus on change in the network.

Apart from these, learning in ANN is classified into three categories as

i) supervised learning

ii) unsupervised learning

Iii) reinforcement learning

Supervised learning

In ANN, each input vector requires a corresponding target vector, which represents the desired output.

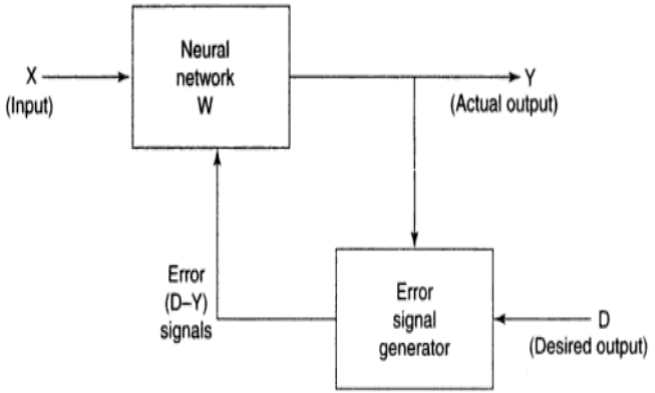
The input vector along with target vector is called ***training pair.***

The input vector results in output vector.

The actual output vector is compared with desired output vector.

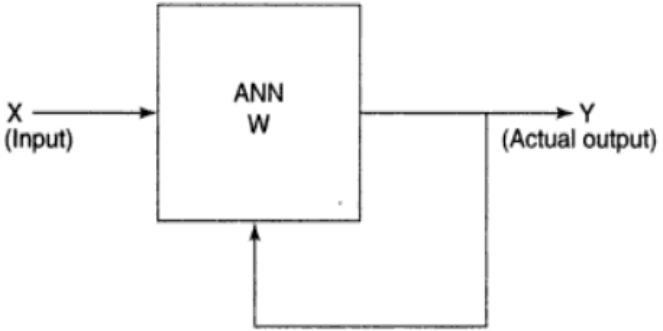
If there is a difference means an error signal is generated by the network.

It is used for adjustment of weights until actual output matches desired output.



Unsupervised learning

- Learning is performed without the help of a teacher.
- Example: tadpole – learn to swim by itself.
- In ANN, during training process, network receives input patterns and organize it to form clusters.
- From the Fig. it is observed that no feedback is applied from environment to inform what output should be or whether they are correct.
- The network itself discover patterns, regularities, features/ categories from the input data and relations for the input data over the output.
- Exact clusters are formed by discovering similarities & dissimilarities so called as *self – organizing.*



Reinforcement learning

- Similar to supervised learning.
- For example, the network might be told that its actual output *is* only "50% correct" or so. Thus, here only critic information is available, nor the exact information.
- Learning based on *critic information* is called *reinforcement learning* & the feedback sent is called *reinforcement signal.*
- The network receives some feedback from the environment.
- Feedback is only evaluative.

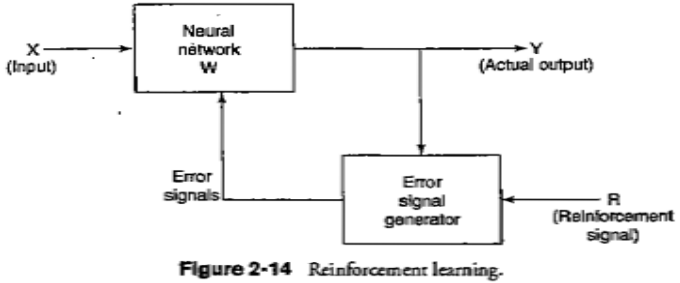
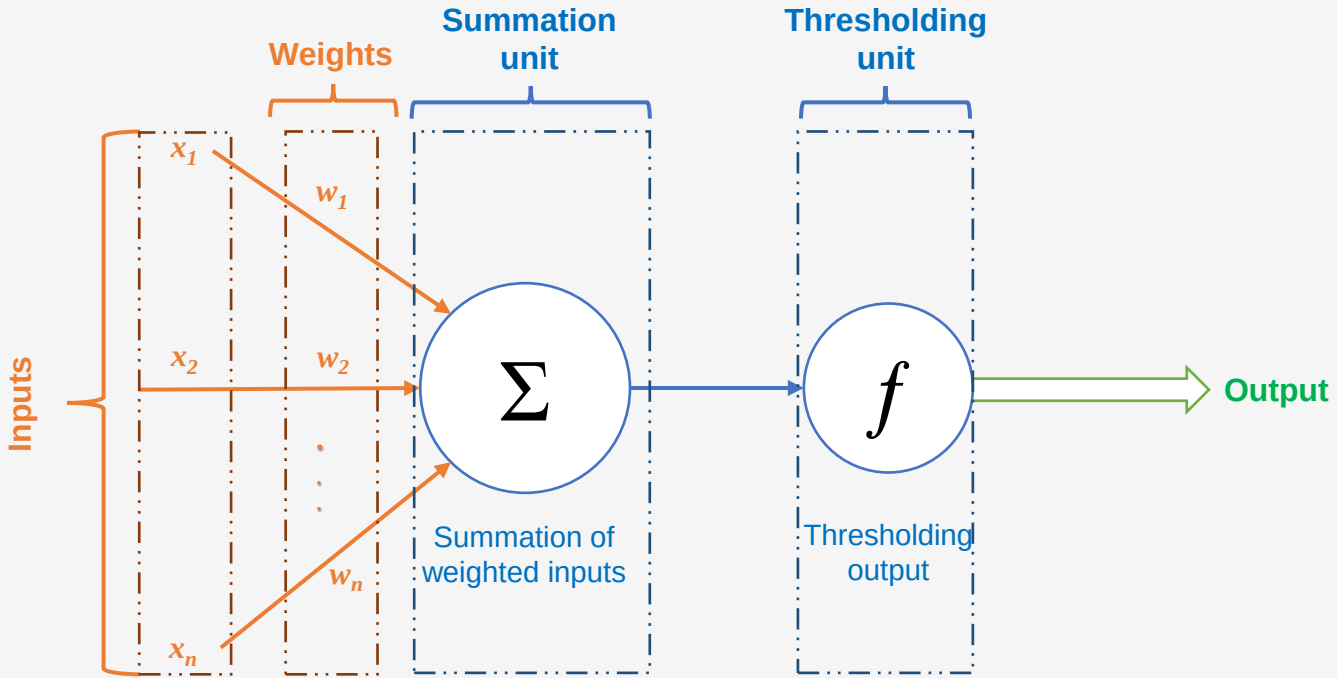


Figure 2-14 Reinforcement learning.

The external reinforcement signals are processed in the critic signal generator, and the obtained critic signals are sent to the ANN for adjustment of weights properly to get critic feedback in future.

Simple Model of Artificial Neuron



Activation functions

- To make work more efficient and for exact output, some force or activation is given.
- Activation function is applied over the net input to calculate the output of an ANN. Information processing of processing element has two major parts: input and output. An integration function (f) is associated with input of processing element.

Simple Model of Artificial Neuron

Let I be the total input received by the soma of the artificial neuron

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

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

To generate the output y, the sum I is passed on to a non-linear filter f called the *Activation function* or *Transfer function* or *Squash Function*

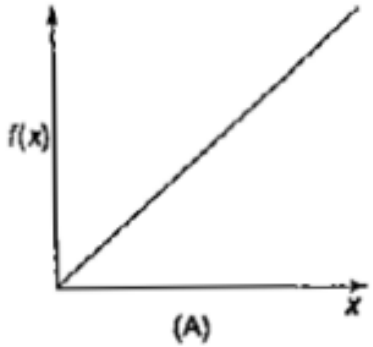
$$y = f(I)$$

Linear Activation Functions

Activation functions-Identity function

1. **Identity function(Linear Activation fucntion):**
   - it is a linear function which is defined as

     f(x) =x for all x
   - The output is same as the input.



Linear Activation Function(Identity function)

- **Range**: -infinity to +infinity
- **Uses**: Used only in output layer
- Limitation: problems with linear activation function
    - Derivative of function is constant and has no relation with the input, so not suitable for backpropogation
    - All layers in the network will collapse into one layer. No matter the number of layers, output layer will be linear function of first layer
    - Not able to learn complex patterns from the data.

Activation Functions: *Heaviside function/Step function*

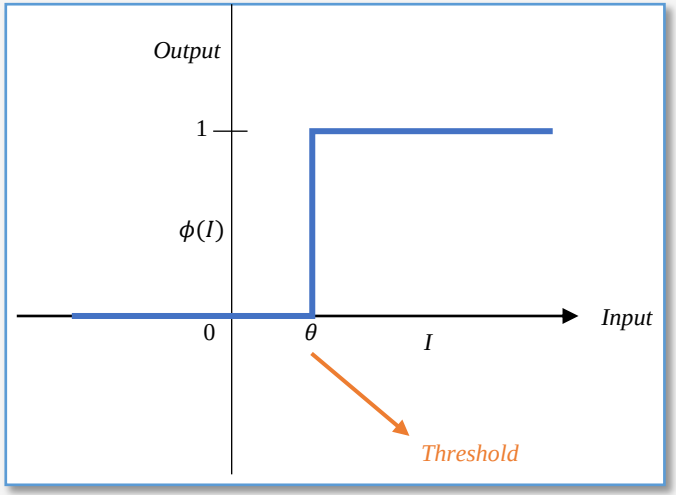
Very commonly used activation function: Thresholding function

The sum is compared with a threshold value θ. If $I > \theta$, then the output is *1* else it is *0*

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

where, ϕ is the step function known as Heaviside function and is such that

$$\mathrm{f}(I) = \begin{cases} 1, & I > 0 \\ 0, & I \leq 0 \end{cases}$$



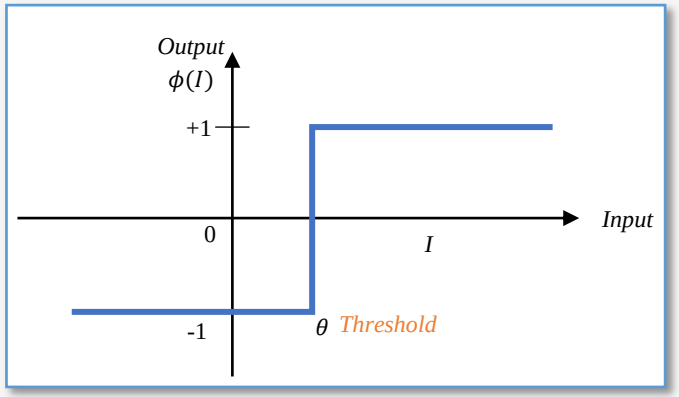
Binary Step function (Unipolar Binary)

- **Range**: 0 or 1
- **Uses**: Used for binary classification (yes or no) problem
- **Limitation**: problems with linear activation function
  - Cannot be used for multi-class classification problem.
  - Gradient of this function is zero, which causes problem in back propagation process.
  - Not able to learn complex patterns from the data.

Activation Functions: *Signum function (Biplolar Step function*

Also known as Quantizer function

$$f(I) = \begin{cases} +1, & I > 0 \\ -1, & I \leq 0 \end{cases}$$



Bipolar binary Function

- Range: -1 or 1
- Uses: Used for binary classification (yes or no) problem
- Limitation: 2 problems with linear activation function
  - Cannot be used for multi-class classification problem.
  - Gradient of this function is zero, which causes problem in back propagation process.
  - Not able to learn complex patterns from the data.

Non-Linear Activaiton Functions

Activation Functions: *Sigmoidal function*

Varies gradually between the asymptotic values 0 and 1 (logistic function)

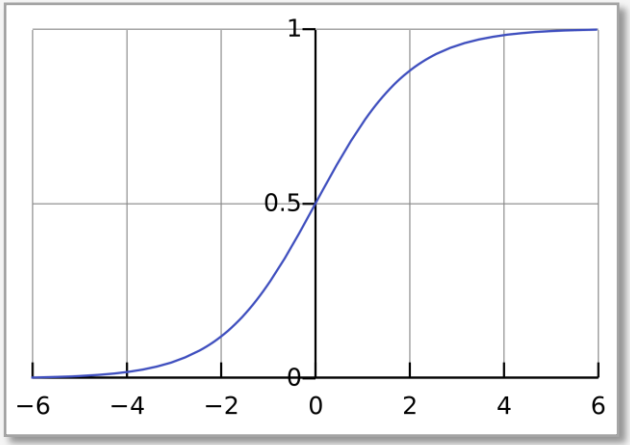
$$f(I) = \frac{1}{1+e^{-\alpha I}}$$

where, α is the slope parameter

The function is differentiable

Prone to vanishing gradient problem

When gradient reaches 0, the network do not learn



Activation Functions: *Sigmoidal function*

- **Range**: 0 to 1
- Function takes real value as input and output values in range of 0 to 1.
- Larger the input , output value is close to 1 whereas smaller the input, the output is close to 0
- **Uses**: Used to predict the probability of output (probability lies bwn 0 to 1)
- Function is differentiable.
- **Limitation**:
    - Suffers from **vanishing gradient problem** (When inputs are in the saturated regions of these functions (very high or very low values), the derivatives are close to zero. During backpropagation, gradients are calculated as the product of these derivatives. If many derivatives are close to zero, the gradient diminishes exponentially as it is propagated backward through each layer) . Network doesn't learn

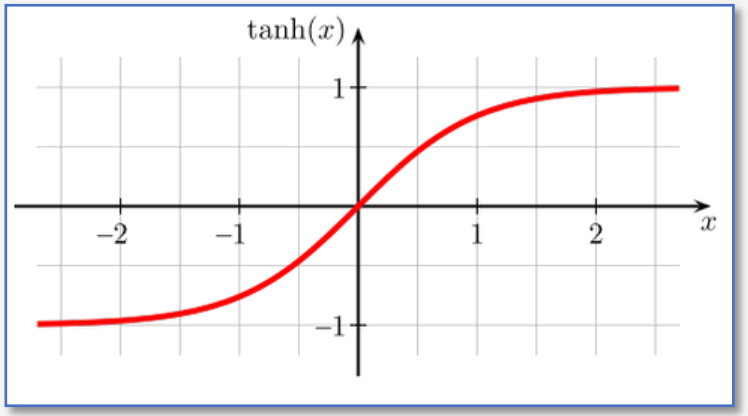
Activation Functions: *Hyperbolic tangent function*

Also known as *tanh* function

$\mathrm{f}(I) = \tanh(I)$ $f(x) = \dfrac{2}{1+e^{-\lambda x}} - 1 = \dfrac{1-e^{-\lambda x}}{1+e^{-\lambda x}}$

Scaled version of sigmoid function

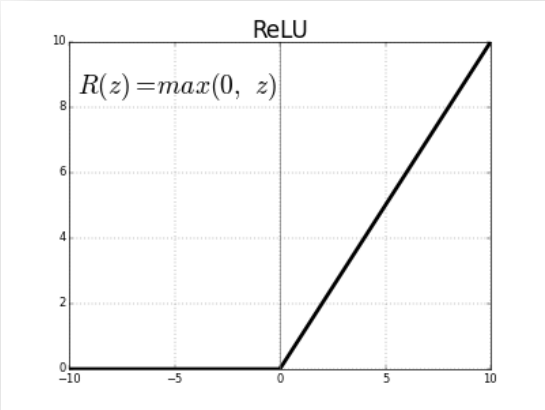
Leads to vanishing gradient problem in very deep neural networks



Activation Functions: Bipolar *Sigmoidal function/tanh*

- **Range**: -1 to 1
- Function takes real value as input and output values in range of -1 to 1.
- Output of tanh function is zero centered.
- **Uses**: Usually used in hidden layers.
- Function is differentiable.
- **Limitation**:
    - Suffers from **vanishing gradient problem like sigmoid but tanh is zero centered** and gradient are not restricted to move in certain direction. Therefore tanh non-linearity and more prefered to sigmoid non-linearity

Other popular activation functions: ReLU (Rectified linear unit) and Softmax



- Most widely used
- Does not activate all neurons at the same time
- If input is negative the neuron will not get activated
- Overcomes the vanishing gradient problem
- Suited for hidden layers

Softmax Function

Softmax is a type of sigmoid function

Used in handling

Ideally used in output layer of the classification

$$I_n = \frac{e^{z_n}}{\sum_{k=1}^{m} e^{z_k}}$$

Activation Functions: ReLU

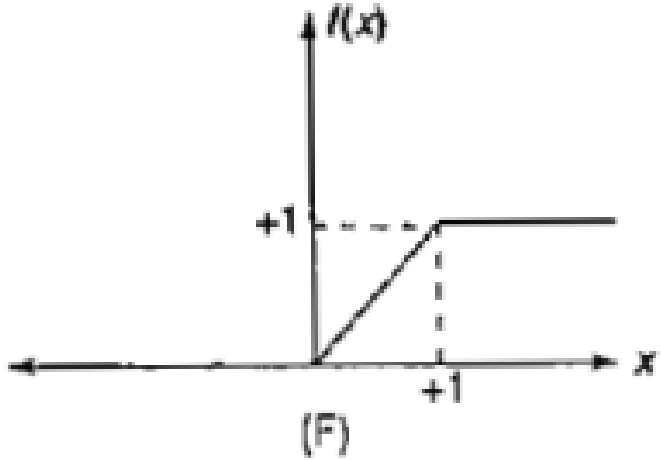
- **Range**: 0 to infinity
- Relu function does not activate all neurons at the same time.
- **Uses**: Computational inexpensive than tanh and sigmoid
  - Learns much faster than sigmoid and tanh
  - Mostly implemented in hidden layers of NN
- Function is differentiable.
- **Limitation**:
  - Dying ReLU problem. All negative values are convereted to zero and this conversion rate is so fast that neither it can map nor fit into data properly which creates a problem.

Activation function: SoftMax

- Uses: Usually used when trying to handle multiple class. It is often used as activation function of NN to normalize the output of a network to a probability distribution over predicted output class.

Ramp function:

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



(F)

Properties of Good activation function

Zero Centered: Output of activation is zero centered, than gradient do not shift in one direction

Computational expense: Computationaly inexpensive as the activation function is applied after every layer in calculated million number of times in deep network.

Differentiable: Neural network are trained using gradient descent process, hence the layers in the network should be differentiable or nearly differentiable. Hence this requirement for activation function.

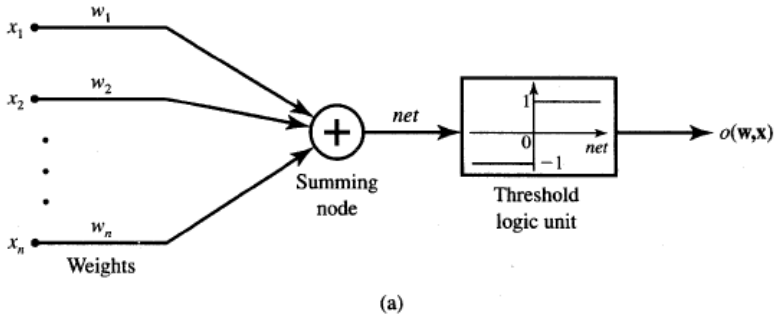
Range: Range of the values generated by activation function is imp factor for its application

Properties of Good activation function

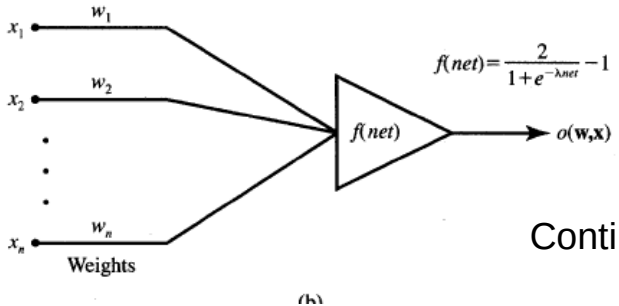
Robust to vanishing gradient problem: NN are trained using gradient descent process. Gradient descent will involve backpropagation steps where the weights are updated to reduce the loss of each epoch. The activation function should withstand vanishing gradient.

Non-linearity- Non-linear activation function are preferred over linear activation function to solve complex problem.

Common models of neurons



Binary perceptrons



Continuous perceptrons

Weights

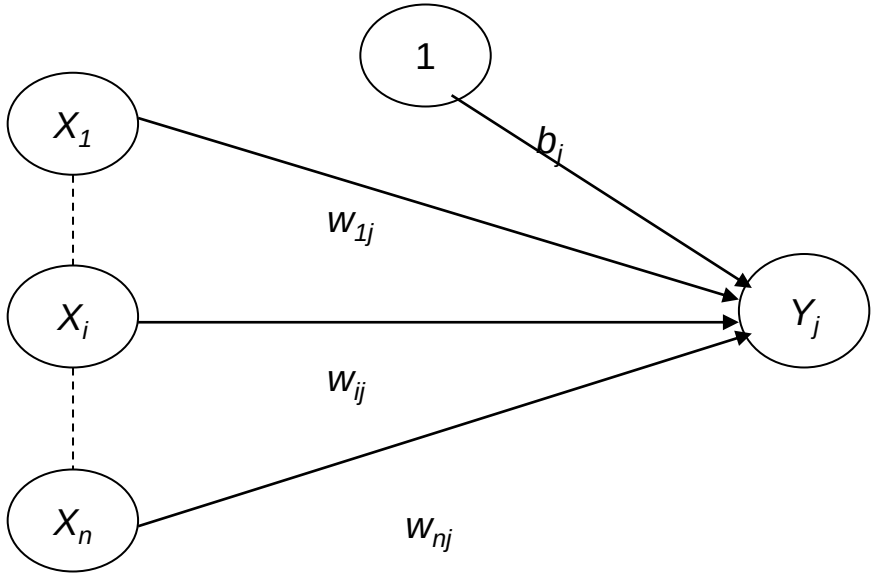
- Each neuron is connected to every other neuron by means of directed links
- Links are associated with weights
- Weights contain information about the input signal and is represented as a matrix
- Weight matrix also called connection matrix

Weight matrix

$$W = \begin{bmatrix} w_1^T \\ w_2^T \\ w_3^T \\ . \\ . \\ . \\ . \\ . \\ w_n^T \end{bmatrix} = \begin{bmatrix} w_{11} w_{12} w_{13} \cdots w_{1m} \\ w_{21} w_{22} w_{23} \cdots w_{2m} \\ \cdots\cdots\cdots\cdots\cdots \\ \cdots\cdots\cdots\cdots\cdots \\ w_{n1} w_{n2} w_{n3} \cdots w_{nm} \end{bmatrix}$$

Weights contd...

- w_{ij} _is the weight from processing element *"i"* (source node) to processing element *"j"* (destination node)



$$y_{inj} = \sum_{i=0}^{n} x_i w_{ij}$$

$$= x_0 w_{0j} + x_1 w_{1j} + x_2 w_{2j} + ... + x_n w_{nj}$$

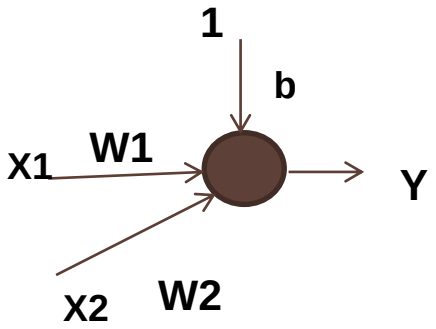
$$= w_{0j} + \sum_{i=1}^{n} x_i w_{ij}$$

$$y_{inj} = b_j + \sum_{i=1}^{n} x_i w_{ij}$$

Bias

- Bias has an impact in calculating net input.
- Bias is included by adding x_0 to the input vector x.
- The net output is calculated by

$$y_{inj} = \sum_{i=0}^{n} x_i w_{ij} \qquad y_{inj} = b_j + \sum_{i=0}^{n} x_i w_{ij}$$



- The bias is of two types
  - Positive bias -Increase the net input
  - Negative bias-Decrease the net input

Threshold

- It is a set value based upon which the final output is calculated.
- Calculated net input and threshold is compared to get the network output.
- The activation function of threshold is defined as

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

where θ is the fixed threshold value

Learning rate

- Denoted by α.
- Used to control the amount of weight adjustment at each step of training
- Learning rate ranging from 0 to 1 determines the rate of learning in each time step

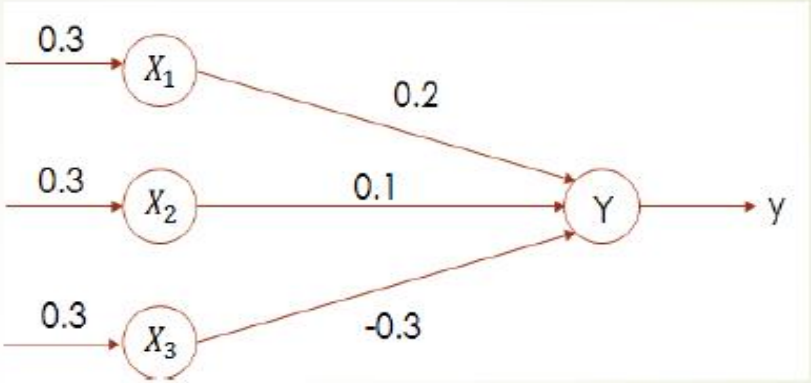
Other terminologies

- Momentum factor:
  - used for convergence when momentum factor is added to weight updation process.
- Vigilance parameter:
  - Denoted by ρ
  - Used to control the degree of similarity required for patterns to be assigned to the same cluster

Module 1.2 (part 2) MP Neuron

Example of NN to calculate net output

Q1. For the network shown in Figure 1, calculate the weights are net input to the output neuron.



Solution: The given neural net consists of three input neurons and one output neuron. The inputs and weights are

$$[x_1, x_2, x_3] = [0.3, 0.5, 0.6]$$

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

The net input can be calculated as

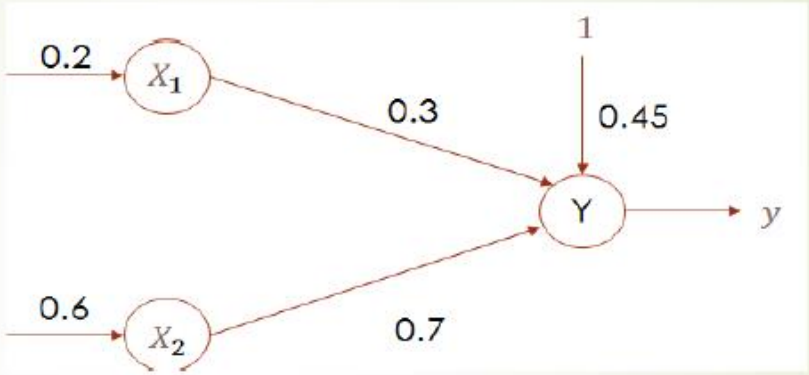
$$y_{in} = x_1 w_1 + x_2 w_2 + x_3 w_3$$

$$= 0.3 \times 0.2 + 0.5 \times 0.1 + 0.6 \times (-0.3)$$

$$= 0.06 + 0.05 - 0.18 = -0.07$$

Example of NN to calculate net output

Q2. Calculate the net input for the network shown in Figure 2 with bias included in the network.



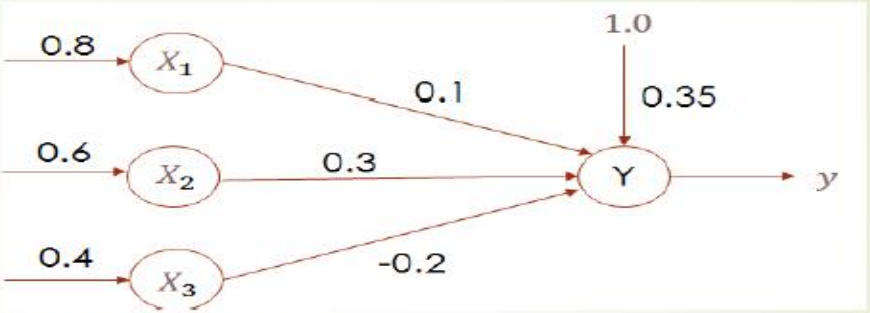
Solution: The given net consists of two input neurons a bias and an output neuron. The inputs are $[x_1, x_2] = [0.2, 0.6]$ and the weights are $[w_1, w_2] = [0.3, 0.7]$. Since the bias is included b=0.45 and bias input x_0 is equal to 1. The net input is calculated as

$$y_{in} = b + x_1 w_1 + x_2 w_2$$
$$= 0.45 + 0.2 \times 0.3 + 0.6 \times 0.7$$
$$= 0.45 + 0.06 + 0.42 = 0.93$$

Therefore, $y_{in} = 0.93$ is the net input.

Example of NN to calculate net output

Q3. Obtain the output of the neuron Y for the network shown in Figure 3 using activation function as: (1)binary sigmoidal and (ii) bipolar sigmoidal.



Solution: The given network has three neurons with bias and output neuron. These form a single layer network. The inputs ere given as $[x_1, x_2, x_3] = [0.8, 0.6, 0.4]$ and the weights are $[w_1, w_2, w_3] = [0.1, 0.3, -0.2]$ with bias b=0.35 (its input is always 1). The net input to the output neuron is

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

[n=3, because only 3 input neurons are given]

$$= b + x_1 w_1 + x_2 w_2 + x_3 w_3$$
$$= 0.35 + 0.8 \times 0.1 + 0.6 \times 0.3 + 0.4 \times (-0.2)$$
$$= 0.35 + 0.08 + 0.18 - 0.08 = 0.53$$

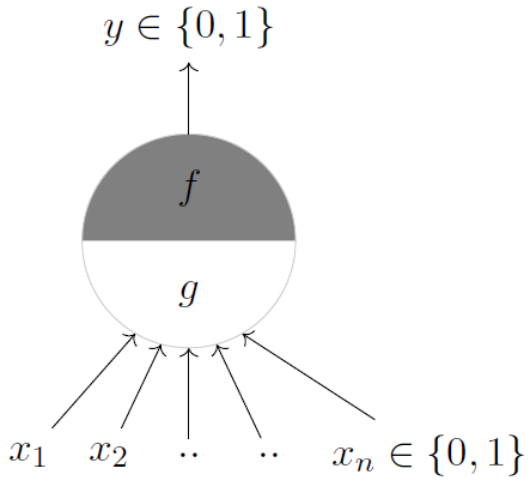
(i)For binary sigmoidal activation function,

$$y = f(y_{in}) = \frac{1}{1+e^{-y_{in}}} = \frac{1}{1+e^{-0.53}} = 0.625$$

(ii) For bipolar sigmoidal activation function,

$$y = f(y_{in}) = \frac{2}{1+e^{-y_{in}}} - 1 = \frac{2}{1+e^{-0.53}} - 1 = 0.259$$

McCulloch Pitts Neuron (MP neuron)



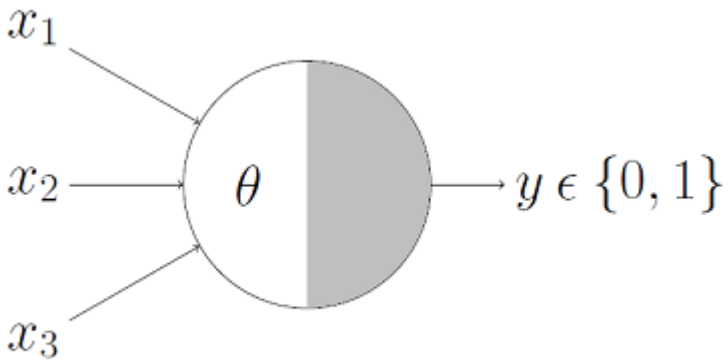
- McCulloch (neuroscientist) and Pitts (logician) proposed a highly simplified computational model of the neuron (1943)
- g aggregates the inputs and the function f takes a decision based on this aggregation
- The inputs can be excitatory or inhibitory
- $y = 0$ if any x_i is inhibitory, else

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

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

- θ is called the thresholding parameter
- This is called Thresholding Logic

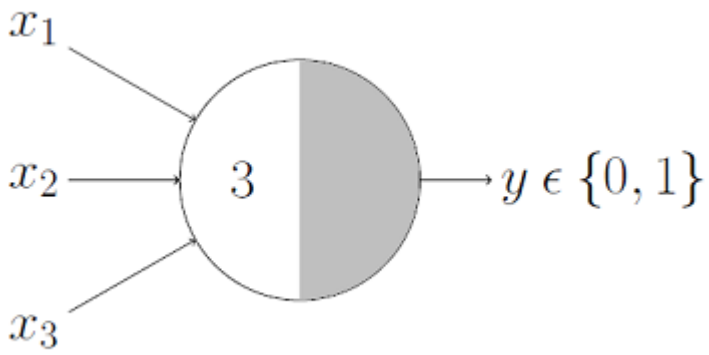
Boolean Functions Using M-P Neuron



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

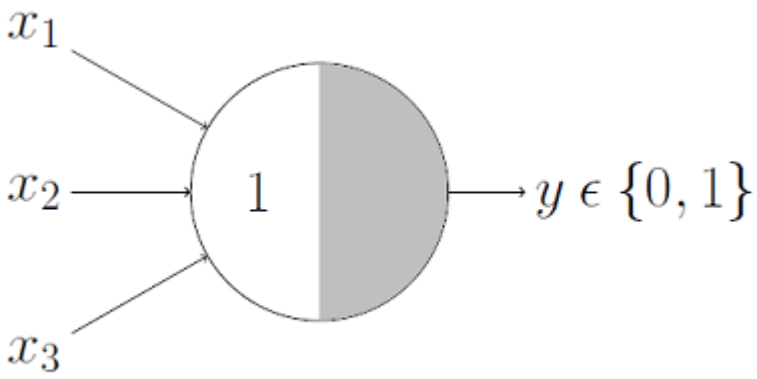
Concise representation of M-P neuron

AND Function



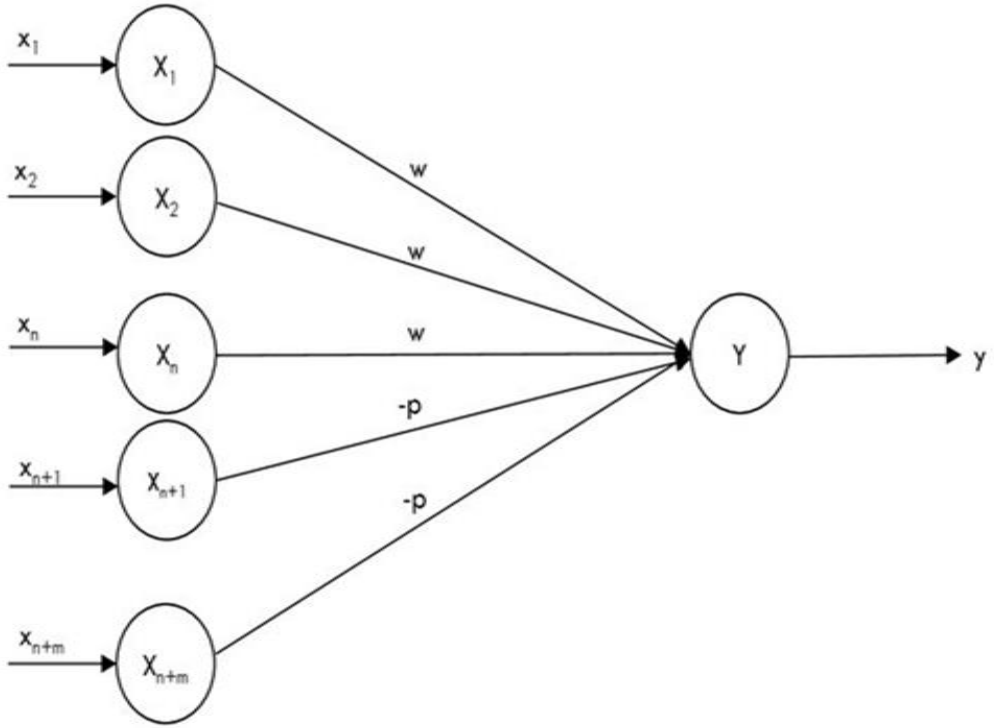
AND function neuron would only fire when ALL the inputs are ON i.e., ***g*(x)** ≥ 3 .

OR Function



- OR function neuron would fire if ANY of the inputs is ON i.e., g(x) ≥ 1

McCULLOCH-PITTS NEURON: ARCHITECTURE (MP NEURON MODEL)

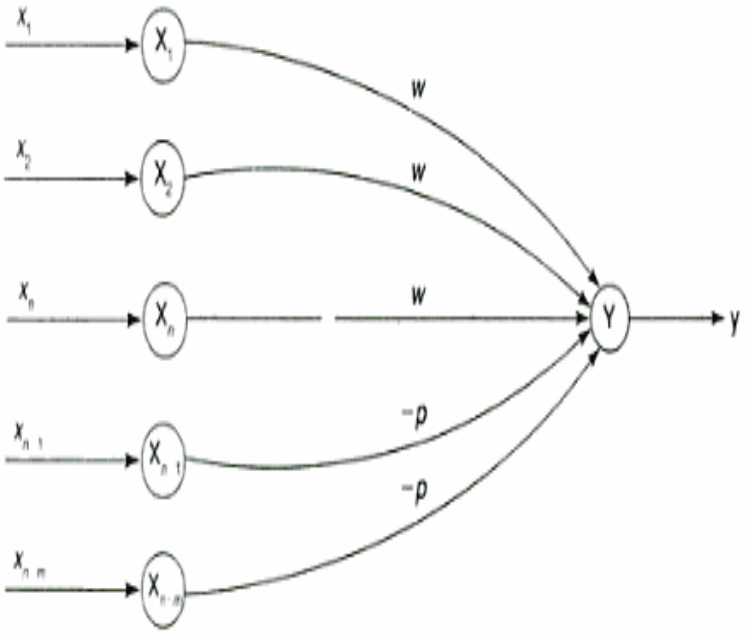


McCULLOCH-PITTS NEURON

Activation function for MP neuron is

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

And $\theta > nw - p$



- A simple M-P neuron is shown in the figure.
- It is excitatory with weight (*w>0)*/inhibitory with weight *–p (p<0)*.
- In the Fig., inputs from x_1 to x_n possess excitatory weighted connection and X_{n+1} to x_{n+m} has inhibitory weighted interconnections.
- Since the firing of neuron is based on threshold, activation function is defined as

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

- For inhibition to be absolute, the threshold with the activation function should satisfy the following condition:

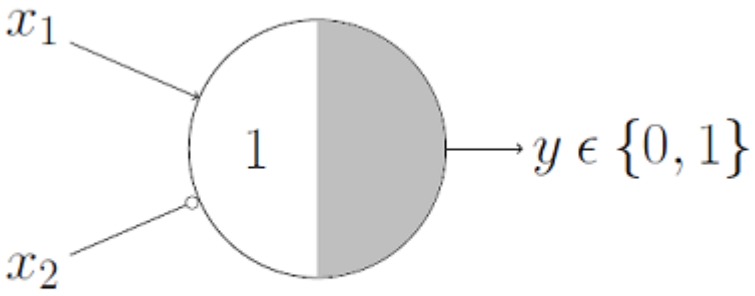
$$\theta > nw - p$$

- Output will fire if it receives "*k*" or more excitatory inputs but no inhibitory inputs where

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

- The M-P neuron has no particular training algorithm.
- An analysis is performed to determine the weights and the threshold.
- It is used as a building block where any function or phenomenon is modeled based on a logic function.

A Function With An Inhibitory Input



$x_1\ AND\ !x_2$*

- we have an inhibitory input i.e., x_2 so whenever x_2 is 1, the output will be 0.
- x_1 ***AND !*** x_2 would output 1 only when x_1 is 1 and x_2 is 0

Computation of threshold for logical AND-NOT operation

6.

Case 1: Assume that both weights w_1 and w_2 are excitatory, i.e.,

$$w_1 = w_2 = 1$$

Then for the four inputs calculate the net input using

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

For inputs

$$(1, 1),\ y_{in} = 1 \times 1 + 1 \times 1 = 2$$

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

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

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

From the calculated net inputs, it is not possible to fire the neuron for input (1, 0) only. Hence, these weights are not suitable.

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

$$w_1 = 1,\ w_2 = -1$$

Now calculate the net input. For the inputs

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

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

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

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

From the calculated net inputs, now it is possible to fire the neuron for input (1, 0) only by fixing a threshold of 1, i.e., $\theta \geq 1$ for Y unit. Thus,

$$w_1 = 1; w_2 = -1;\ \theta \geq 1$$

Note: The value of θ is calculated using the following:

$$\theta \geq nw - p$$

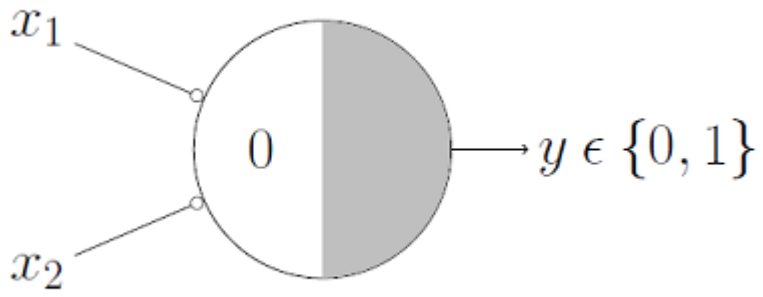
$$\theta \geq 2 \times 1 - 1 \quad \text{[for "}p\text{" inhibitory only magnitude considered]}$$

$$\theta \geq 1$$

Thus, the output of neuron Y can be written as

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

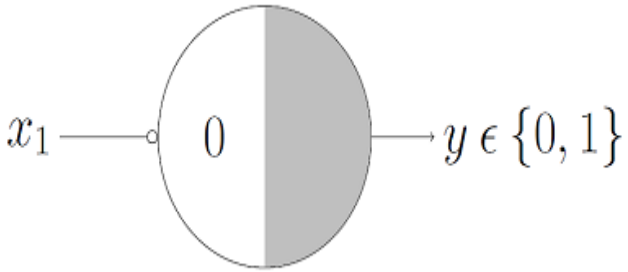
NOR Function



For a NOR neuron to fire, we want ALL the inputs to be 0 so the thresholding parameter should also be 0 and we take them all as inhibitory input

Computation of threshold for logical NOR operation (try)

NOT Function

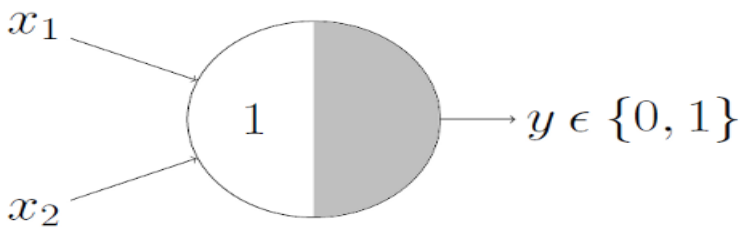


For a NOT neuron, 1 outputs 0 and 0 outputs 1. So we take the input as an inhibitory input and set the thresholding parameter to 0

Computation of threshold for logical NOT operation(try)

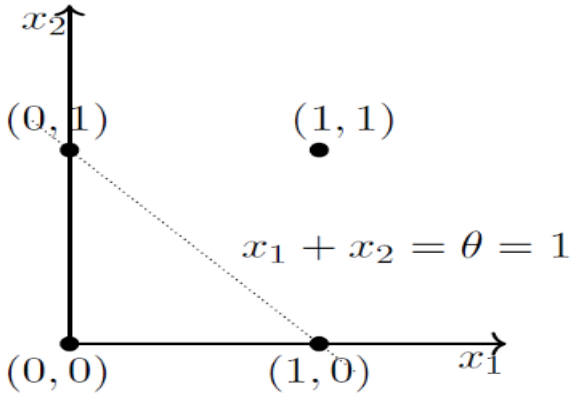
Design an NN using only MP-neuron for NAND (2 inputs) function (try!)

Geometrical interpretation of OR function MP-neuron



OR function

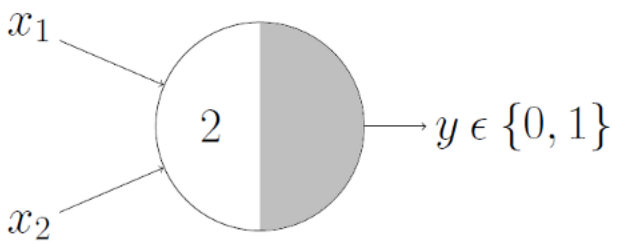
$$x_1 + x_2 = \sum_{i=1}^{2} x_i \geq 1$$



- A single MP neuron splits the input points (4points for 2 binary inputs) into two halves
- Points lying on or above the line $\sum_{i-1}^{n} x_i$-θ =0 and points lying below this line.

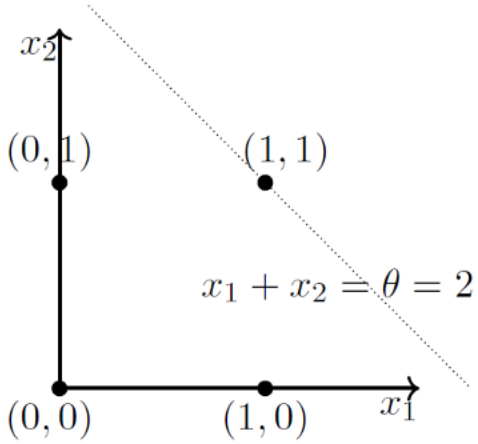
- all inputs which produce an output 0 will be on one side ($\sum_{i=1}^{n} x_i$< θ) of the line and all inputs which produce an output 1 will lie on the other side ($\sum_{i=1}^{n} x_i$> θ) of this line

Geometrical interpretation of AND function MP-neuron

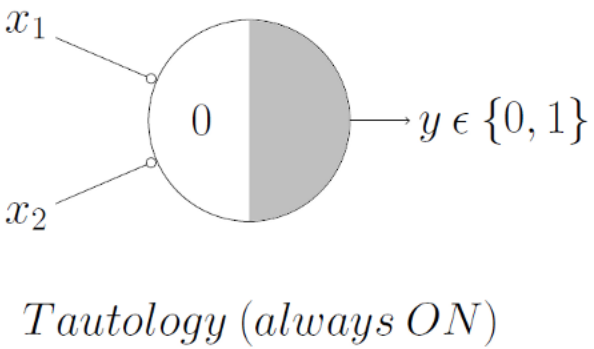


AND function

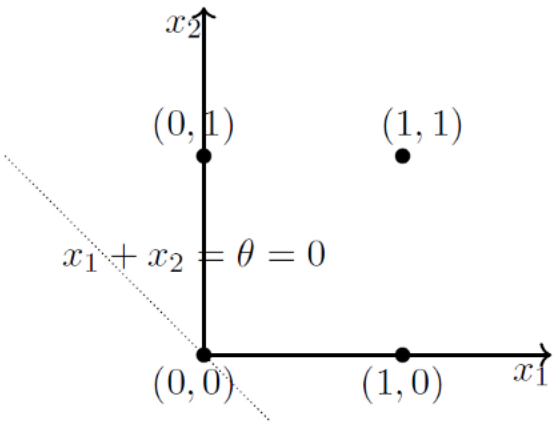
$$x_1 + x_2 = \sum_{i=1}^{2} x_i \geq 2$$



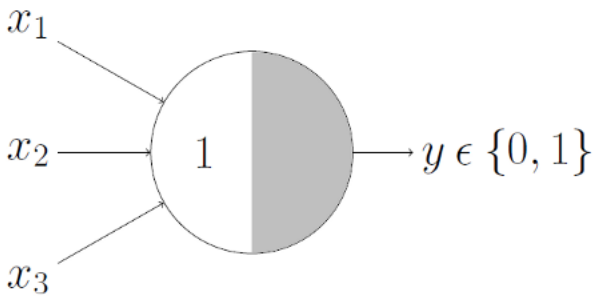
Geometrical interpretation of Tautology MP-neuron



Tautology (always ON)

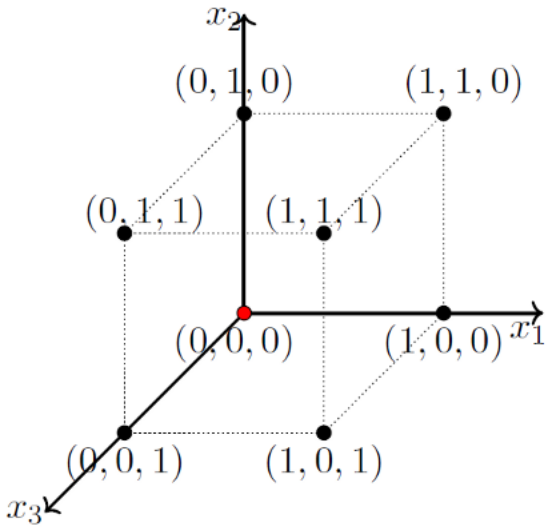


Geometrical interpretation of OR function MP-neuron with 3 inputs

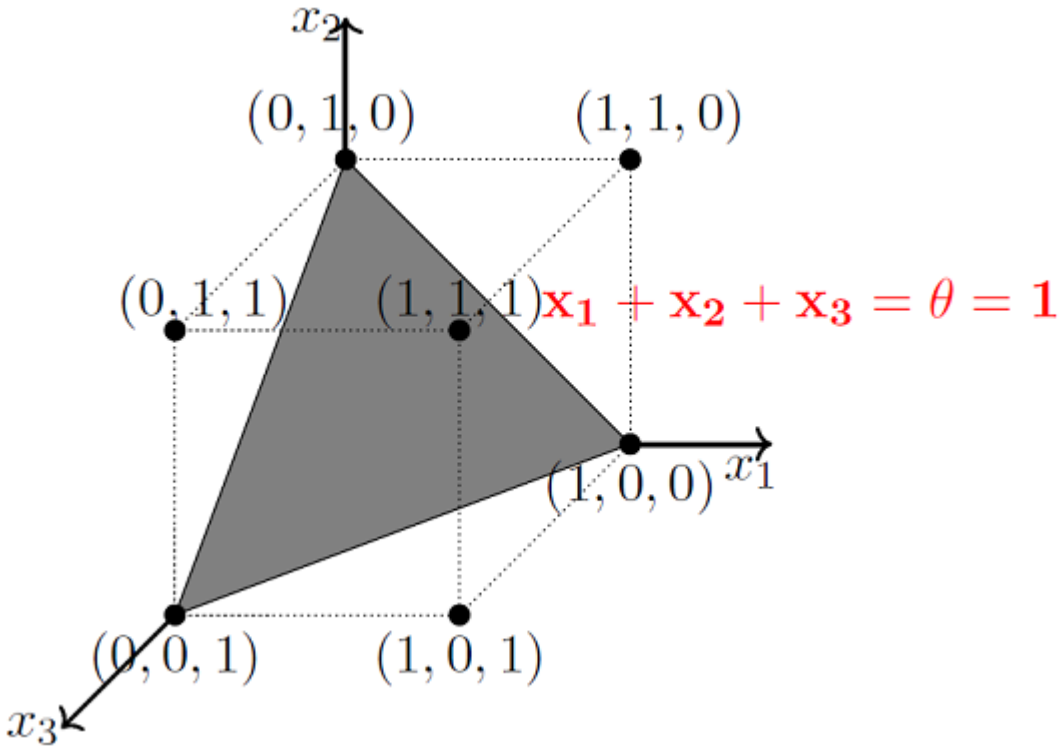


OR function

$$x_1 + x_2 + x_3 = \sum_{i=1}^{3} x_i \geq 1$$



Geometrical interpretation of OR function MP-neuron with 3 inputs

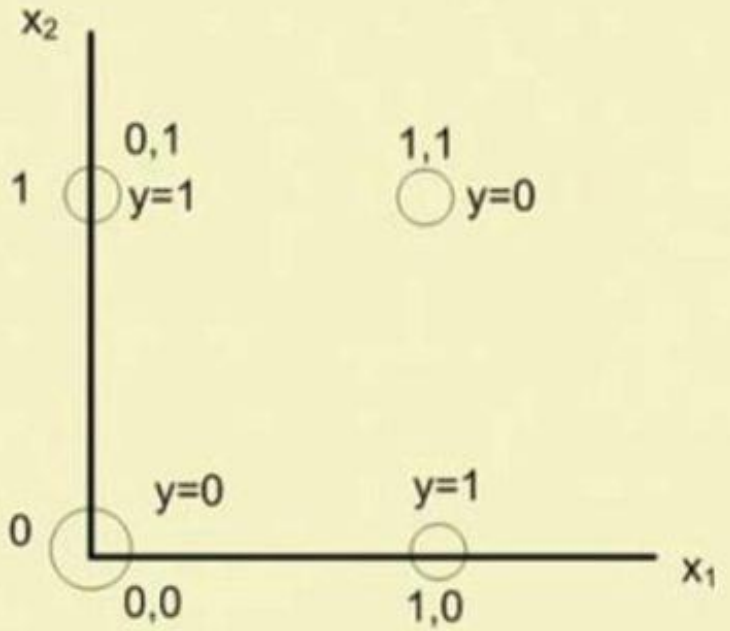


For the OR function, we want a plane such that the point (0,0,0) lies on one side and the remaining 7 points lie on the other side of the plane

XOR PROBLEM IS LINEARLY NON-SEPARABLE

x_1	x_2	Output (y)
0	0	0
0	1	1
1	0	1
1	1	0

XOR Problem

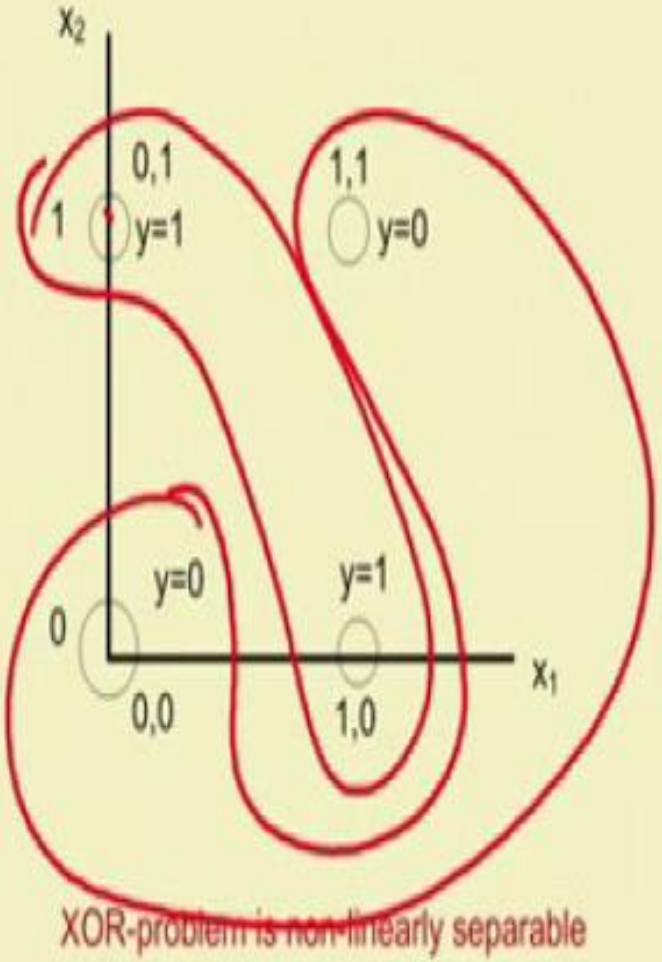


XOR-problem is non-linearly separable

XOR problem is linearly non-separable

x_1	x_2	Output (y)
0	0	0
0	1	1
1	0	1
1	1	0

XOR Problem



XOR-problem is non-linearly separable

Thus..

- A single McCulloch Pitts Neuron can be used to represent boolean functions which are linearly separable
- **Linear separability (for boolean functions)** : There exists a line (plane) such that all inputs which produce a 1 lie on one side of the line (plane) and all inputs which produce a 0 lie on other side of the line (plane)

Limitation of MP-neuron

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

Linear separability

- It is a concept wherein the separation of the input space into regions is based on whether the network response is positive or negative.
- A decision line is drawn to separate positive or negative response.
- The decision line is also called as *decision-making line or decision-support line or linear-separable line.*
- The net input calculation to the output unit is given as

$$y_{in} = b_j + \sum_{i=1}^{n} xiwi$$

- The region which is called as *decision boundary* is determined by the relation $b + \sum_{i=1}^{n} xiwi = 0$

Contd..

- Linear separability of the network is based on decision-boundary line.
- If there exists weights(bias) for which training input vectors have positive response (+1 ve) input lie on one side of line and other having negative response(-ve) lie on other side of line, we conclude the problem is linearly separable

Contd..

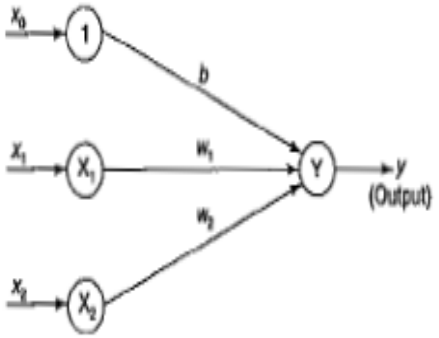


Figure 2-19 A single-layer neural net.

Net input of the network

$$Y_{in} = b + x_1 w_1 + x_2 w_2$$

Separating line for which the boundary lies between the values of x_1 and x_2

$$b + x_1 w_1 + x_2 w_2 = 0$$

If weight of w_2 is not equal to 0 then we get

$$x2 = -w_1/w_2 - b/w_2$$

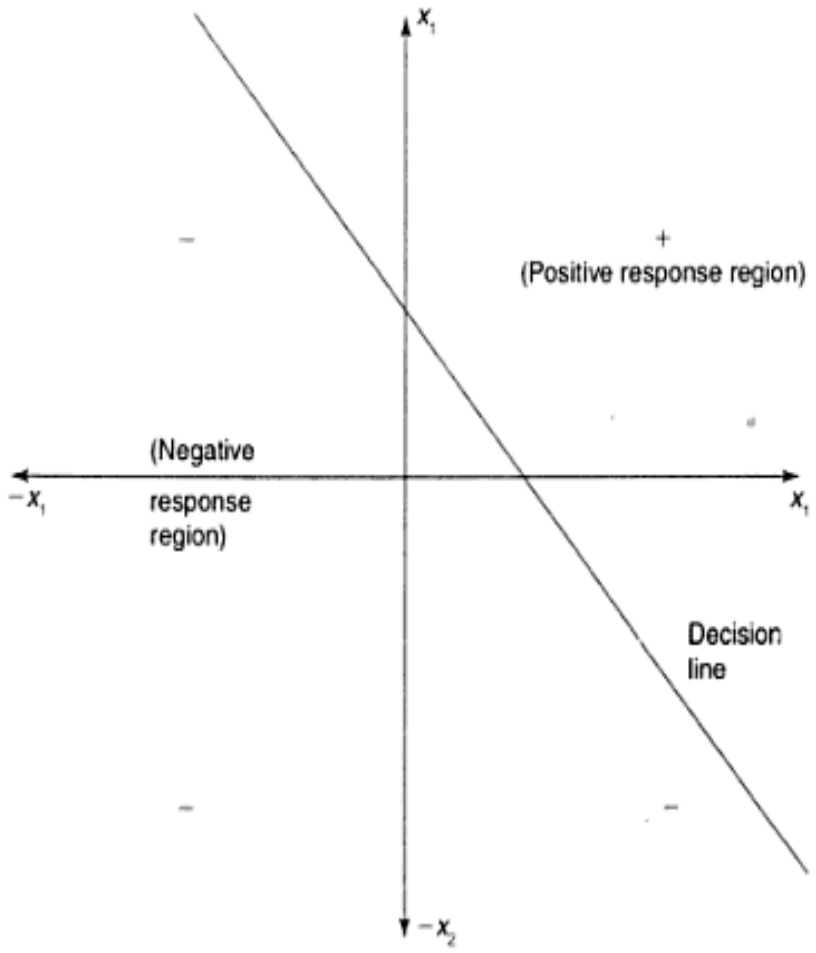
Net requirement for positive response

$$b + x_1 w_1 + x_2 w_2 > 0$$

During training process , from training data w1, w2 and b are decided.

Contd..

- Consider a network having positive response in the first quadrant and negative response in all other quadrants with either binary or bipolar data.
- Decision line is drawn separating two regions as shown in Fig.
- Using bipolar data representation, missing data can be distinguished from mistaken data. Hence bipolar data is better than binary data.
- Missing values are represented by 0 and mistakes by reversing the input values from +1 to -1 or vice versa.



Numericals on linear separability to be added

Hebb network

- In 1949, Donald Hebb said that learning in the brain occurs due to the change in the synaptic gap
- Hebb proposed one of the key ideas in biological learning, commonly known as **Hebb's Law**.
- Hebb's Law states When an axon of cell A is near enough to excite cell B, and repeatedly or permanently takes place in firing it, some growth process or metabolic change takes place in one or both the cells such than A's efficiency, as one of the cells firing B, is increased.

Hebb's Law can be represented in the form of two rules:

1. **If two neurons on either side of a connection are activated synchronously, then the weight of that connection is increased.**

2. **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-Unsupervised learning (Learning here is a **local phenomenon** occurring without feedback from the environment)

Hebb network

- According to Hebb rule, ***the weight vector is found to increase proportionately to the product of the input and the learning signal. Here learning signal is equal to neuron output.***
- In Hebb learning, two interconnected neurons are ‘on’ simultaneously. Weight updating associated with these neurons is increased by modification in the synaptic gap (strength)
- The weight update in Hebb rule is given by
    - W_i**(new) =** w_i **(old)+** x_i y**.**
- It is suited more for bipolar data.

Hebb Network

Hebb Rule: $w_i(new) = w_i(old) + x_i y$

The Algorithm:

Step 0 : Initialize the weights. Basically in this network they may be set to zero, i.e., $w_i = 0 \; for \; i \; to \; n$

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

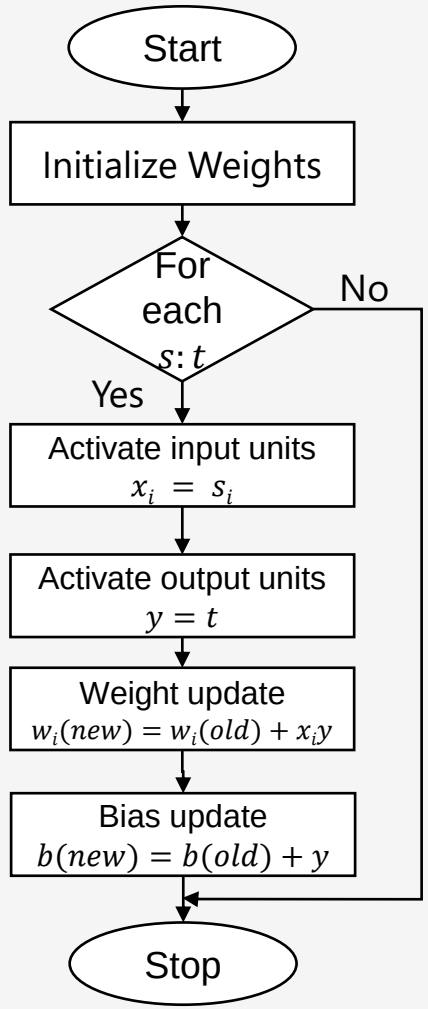
Step 2 : Input units activations are set. Generally, the activation function of the input layer us identify function: $x_i = s_i \; for \; i = 1 \; to \; n$

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

Step 4 : Weight adjustments and bias adjustments are performed:

$$w_i(new) = w_i(old) + x_i y$$

$$b(new) = b(old) + y$$



Design a Hebb net to implement OR function

Use bipolar data in the place of binary data Initially the weights and bias are set to zero w1=w2=b=0

Inputs			Target
X1	X2	B	y
1	1	1	1
1	-1	1	1
-1	1	1	1
-1	-1	1	-1

- **First input [x1 x2 b] = [1 1 1] and target = 1 [i.e. y = 1],** setting the intial weights as

old weights and applying the hebb rule, we get

wi (new) = wi(old) + xi*y

w1(new) = w1(old) + x1*y = 0+1*1 = 1

w2(new) = w2(old) + x2*y = 0+1*1 = 1

b(new) = b(old) +y = 0+1 = 1

- **Similarly for second input [x1 x2 b] = [1 -1 1] and y = 1**

w1(new) = w1(old) + x1*y = 1+1*1 = 2

w2(new) = w2(old) + x2*y = 1+(-1)*1 = 0

b(new) = b(old) +y = 1+1 = 2 & Δb = 1

- **Similarly for third and fourth input, output can be calculated.**

Inputs			y	Weights		
X1	X2	b	Y	W1(0)	W2(0)	b(0)
1	1	1	1	1	1	1
1	-1	1	1	2	0	2
-1	1	1	1	1	1	3
-1	-1	1	-1	2	2	2

Design the Hebb network Numericals

- Design and implement Hebbs Network for OR, AND, NAND function. Use Bipolar inputs and targets
- Design and Implement Hebbs Network for the patter given.