NEURAL NETWORKS AND FUZZY LOGIC

CSE-8 UNIT-1 (DIVYA BHARDWAJ) 48

UNIT-1: Syllabus: Introduction: Concept of Neural Networks, Characteristics of Neural Networks, Historical perspective and application of Neural Networks.

Fundamental of Neural Networks: The Biological prototype, Neuron concept, Single layer Neural Networks, Multi Layer Neural Networks, Terminology, Notation and representation of Neural Networks, Training of Artificial Neural Networks. Representation of Perceptron and issues, Perceptron learning and training, classification, linear separability.

(A) INTRODUCTION: Concept of Neural Networks

→ A neural network is a processing device, either an algorithm or an actual hardware, whose design was inspired by the design and functioning of animal brains and components thereof.

→ The neural networks have the ability to learn by example, which makes them very flexible and powerful.

→ For neural networks, there is no need to devise an algorithm to perform a specific task, i.e there is no need to understand the internal mechanisms of the task.

→ These networks are also well suited for real time systems because of their fast response and computational times which are because of their parallel architecture.

→ ARTIFICIAL NEURAL NETWORKS :

→ An artifical neural network (ANN) may be defined as an information - processing model that is inspired by the way biological nervous systems, such as brain, process information.

→ The model tries to replicate only the most basic functions of the brain.

→ An ANN is composed of large number of highly interconnected processing elements (neurons) working together to solve specific problems.

→ Advantages of ANNs :-

i) Adaptive Learning : An ANN is endowed with the ability to learn how to do tasks based on the data given for training or initial experience.

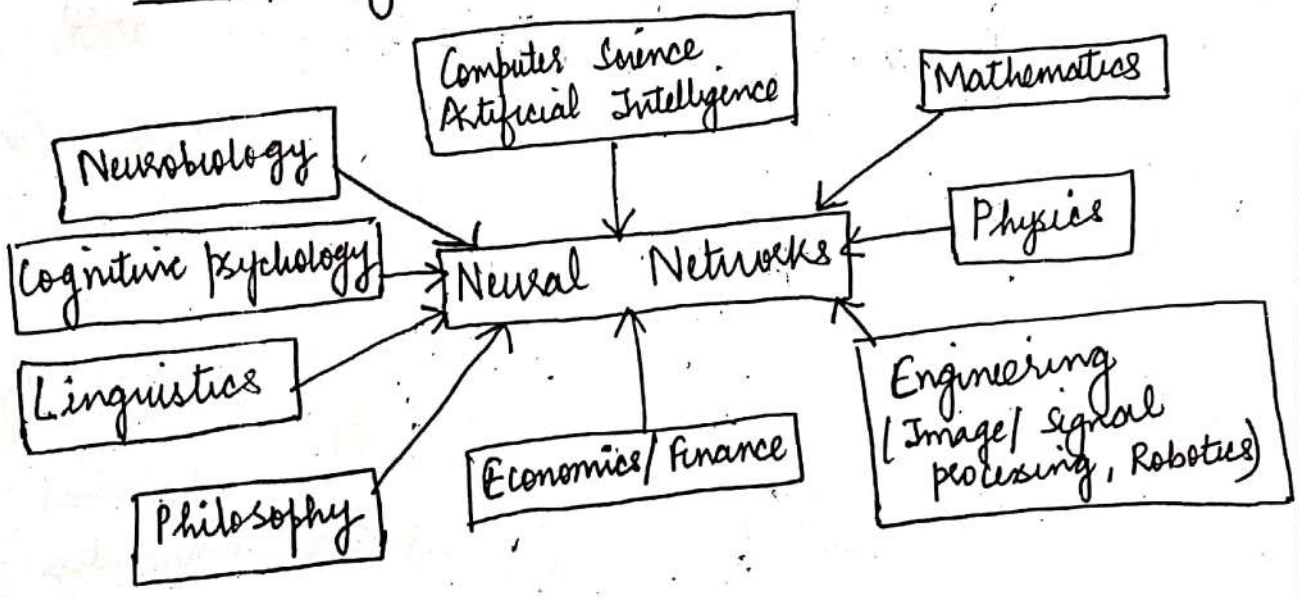
ii) Self - Organization : An ANN can create its own organization or representation of the information it receives during learning time.

iii) Real - time Operation : ANN computations may be carried out in parallel. Special hardware devices

are being designed and manufactured to take advantage of this capability of ANNs.

iv) Fault tolerance via redundant information coding: Partial destruction of a neural network leads to the corresponding degradation of performance. However some network capabilities may be retained even after major network damage.

→ Neural networks can be viewed from a multi-disciplinary point of view.



→ Structure of ANNs:

i) ANN consist of many nodes i.e processing units that are similar to neurons in the brain.

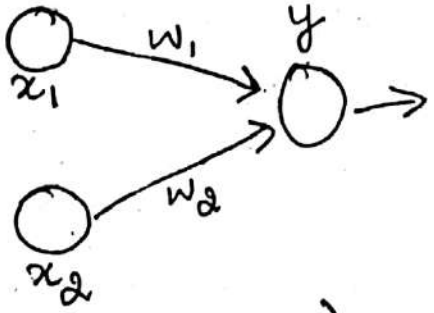
ii) Processing elements (neurons) proces the information.

iii) The signals are transmitted by means of connection links.

iv) The link possess an associated weight, which is multiplied along with the incoming signal.

v) The output signal is obtained by applying activations to the net input.

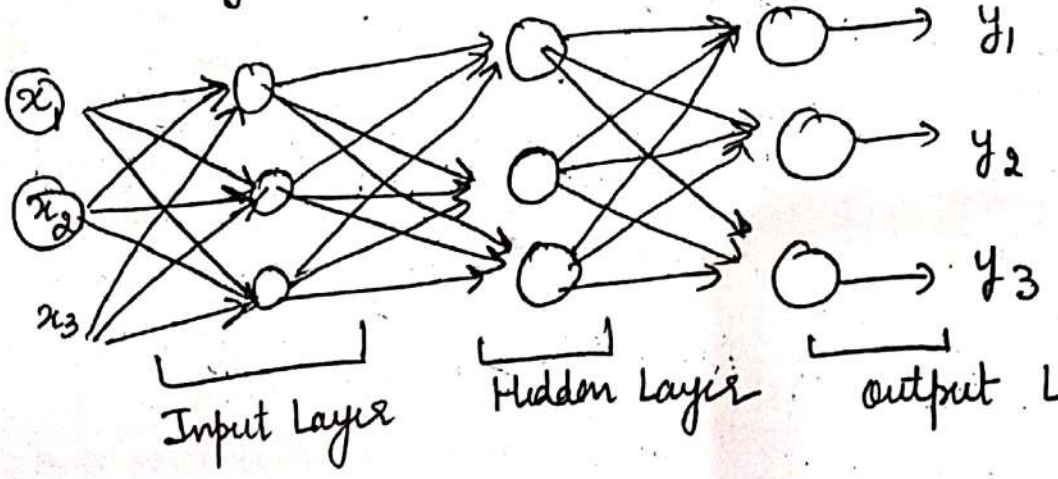
vi) The neural net can generally be a single layer or a multi layer net.



(fig: A simple ANN [Single Layer ANN].

Figure shows a simple ANN with two input neurons (x_1, x_2) and one output neuron (y). Interconnected weights are given by w_1 and w_2.

vii) A typical multi-layer ANN, comprises an input layer, output layer and hidden layer.



(B) CHARACTERISTICS OF NEURAL NETWORKS :-

i) Massive Parallelism : ANNs are massively parallel computing systems consisting of an extremely large number of simple processors with many interconnections.

ii) Learning ability: ANNs can modify their behaviour in response to their environment. According to the set of inputs they receive, they can adjust accordingly to produce consistent responses.

iii) Generalization Ability: Generalization of the ANN is ability to handle unseen data. This means that a trained net could classify data from the same class as the learning data that it has never seen before.

iv) Adaptivity: A neural network trained to operate in a specific environment can be easily retrained to deal with minor changes.

v) Inherent Contextual Information Processing: Knowledge is represented by the structure and activation state of neural network.

vi) Fault-tolerance: A neural network has the potential to be fault tolerant or capable of robust computation.

vii) Distributed Representation and computation:- It means each concept is represented by many neurons. "Distributed Representation" means a many to many relationship between two types of representation (such as concepts & neurons)

viii) Low energy consumption :-
The energy consumed for processing is comparatively low than consumed by conventional circuits.

(c)* APPLICATIONS OF NEURAL NETWORKS :

(1) Data mining, cleaning and validation could be achieved by determining which records suspiciously diverge from the pattern of their peers.

(2) Voice recognition could be obtained by analyzing the oscilloscope pattern.

(3) Medical diagonsis is an ideal application of neural network. (Breast cancer analysis, ECG analysis)

(4) Fraud detection regarding credit cards, insurance or taxes could be automated using a neural network analysis of past incidents.

(5) Business : Neural Networks are being applied in number of business settings. Example are Mortgage Assessment work. Neural Network use past experience to train the network to provide more consistent and reliable evaluation of mortgage apps.

(6) Criminal Sentencing could be predicted using a large sample of crime details as input and the resulting sentences as output.

(7) Traffic flows could be predicted so that signal timing could be optimized.

(8) Weather prediction : Inputs would include - Weather reports from surrounding areas. Output would be the future weather in specific area based on information.

(9) Staff Scheduling requirements for restaurants, retail stores, police stations, banks etc could be predicted based on customer flow, day of week, holidays, season etc.

(10) Photos and finger prints could be recognized by imposing a fine grid over the photo. Each square of the grid becomes an input to neural network.

(11) Scheduling of buses, airplanes and elevators could be optimized by predicting demand.

(12) Music Composition has been tried using neural networks. The network is trained to recognize patterns in the pitch and tempo of certain music and then the network writes its own music.

Other applications are Lake Water Levels prediction, Direct mail advertisers, Retail inventories optimization, Machinery control etc.

① HISTORICAL PERSPECTIVE OF NEURAL NETWORKS :

	Year	Neural Network	Designer	Description
①	1943	MC Culloch and Pitts Neuron	MC Culloch and Pitts	→ A simple logic function is performed by a neuron in this case based upon the weights set in MC-Culloch Pitts Neuron. → The arrangement of neuron in this case may be represented as a combination of logic functions. → The most important feature of this type of neuron is the concept of Threshold. → When the net input to a particular neuron is greater than the spicified threshold by the user, then the neuron fires.
②	1949	HEBB. Network	Hebb	→ The concept behind the Hebb theory is that if two neurons are found to be active simultaneously, the strength of connection between the two neurons should be increased. → This concept is similar to that of the correlation matrix learning.

3	1958	Perceptron	Frank Rosenblatt, Minsky, Papert ~~Widrow Hoff~~	→ The weights on the connection paths can be adjusted. → A method of iterative weight adjustment can be used in perceptron net. → The perceptron net is found to converge if the weights adjustment obtained allow the net to reproduce exactly all the training input and target output vector pairs.
4	1960	Adaline	Widrow and Hoff	→ Adaline abbreviated from Adaptive Linear Neuron uses a learning rule called as Least Mean Square rule or Delta rule. → This rule is found to adjust the weights so as to reduce the difference between the net input to the output unit and desired output. → The convergence criteria in this case are the reduction of mean square error to a minimum value.
5	1972	Kohonen's Self Organizing Maps (SOM)	Kohonen	→ They are capable of reproducing important aspects of the structure of biological neural nets. → They make use of data representation using topographic maps, which are common in nervous systems.

				→ The concept behind this network is that the inputs are clustered together to obtain a fixed output neuron. → These nets are applied to many recognition problems.
(6)	1982	Hopfield Network	John Hopfield and Tank.	→ This neural network is based on fixed weights. → These nets are widely used as associative memory nets. → They can be both continuous valued and discrete valued.
(7)	1986	Back Propagation Network	Rumelhart, Hinton & Williams	→ This network is multi-layer with error being propagated backwards from the output units to the hidden units → Most popular learning algorithms for multilayer perceptron → Used in various neural network applications.
(8)	1988	Counter Propagation Network	Grossberg	→ This network is similar to the Kohonen network → Here the learning occurs for all units in a particular layer → There exists no competition among these units.

	1987-1990	Adaptive Resonance Theory (ART)	Carpenter and Grossberg	→ The ART network is designed for both binary inputs and analog valued inputs. → Here the input patterns can be presented in any order.
16	1988	Radial Basis function Network	Broomhead and Lowe	→ This resembles a back propagation network but the activation function used is a Gaussian function
17	1988	Neo Cognition Network	Fukushima	→ This network is essential for character recognition. → The deficiency occured in cognition network (1975) was corrected by this network.
18	1990	Support Vector Machines	Vapnik	→ It's a new learning machine for two-group classification problems. → They have high generalization ability.

→ Biological Neural Network : (BNN)

- A biological neuron consist of synapses, dendrites, cell body and the axon.
- A neuron is the fundamental building block of the BNN.
- Explanation of the main elements of BNN.

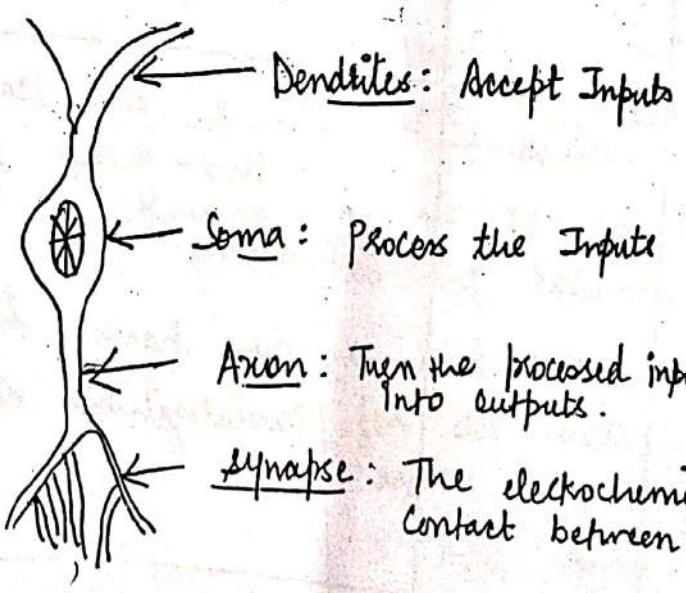
→ Dendrite - Receives signals from other neurons.

→ Soma - Sums of all incoming signals.

→ Axon - When a particular amount of input is received, then cell fires. It is responsible for transmiting signals.

→ Synapses : A synapses is a biochemical device found between neurons. It transmit signal to other neurons.

- Biological Neuron :-



Difference between BNN & ANN

Characteristics	ANN	BNN
Speed:	They are faster in processing information (process in nano sec)	BNN are slow in processing info. (process in millisec)
Processing:	They operate in sequential mode one instruction after another.	BNNs can perform massively parallel operations
Size and Complexity	These do not involve as much computational neurons. Hence it is difficult to perform complex pattern recognition.	The no of neurons in the brain is estimated as 10^{11} and total no of interconnections to be around 10^{15}. The size & complexity give brain the power of performing complex pattern recognition.
Fault Tolerance	They are not inherently fault tolerant.	They exhibit fault tolerance since info is distributed in the connections throughout the network.
Control Mechanism	There is control unit, which monitors all activities of computing.	There is no central control for processing information in the brain.
Storage	Info is stored in memory, which is adressed by its location.	BNN store information in the strength of inter connections.

→ Disadvantages of Neural Network.

1) The Neural Network requires training to operate
2) They requires high processing time for large neural networks.
3) The architecture of neural networks is different from microprocessors so it become hard to interpret the model.

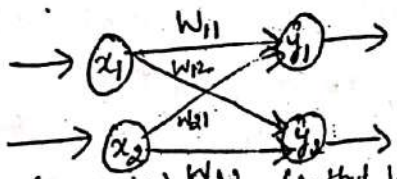
→ Associated Terminologies of Biological and Artificial Neural Net (ANN)

BNN	ANN
Cell Body	Neuron
Dendrite	Weights or interconnections
Soma	Net input
Axon	output

(E) SINGLE LAYER NEURAL NETWORKS Vs MULTI-LAYER NEURAL NETWORKS

Single layer Neural Networks (SLNN)

1. It consists of only one layer of weighted interconnections.
2. The units can be distinguished as input unit, output units
3. Architecture:

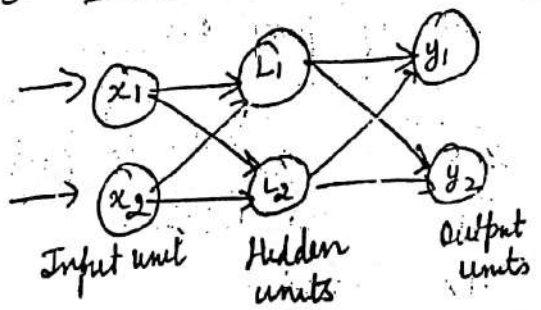


(Single Layer Network with 2 Input units and 2 output units)

4. The Weights from one output unit does not influence the weights for other output units
5. They uses linear activation functions such as Identity function, binary step function

Multi-layer Neural Networks (MLNN)

1. It consist of more than one layer of weighted interconnections.
2. The units can be distinguished as: input units, hidden units, output units.
3. Architecture



(MLNN with 2- Input Units, 2 Hidden Units, 2 output units)

4. The output signals of second layer becomes input for third layer and so on. Hence the weights from one output unit influence the weights for other output units
5. They uses non linear activation functions such as logistic sigmoid.

⑥ Advantages of SLNN:

→ Easy to set up and train.
→ Outputs are weighted sum of inputs: interpretable representation.

⑦. Limitations of SLNN

→ Can represent a limited set of function.

→ It is limited to perform pattern classification with two classes.

⑧. It can be use for classification of patterns that are linear seperable.

Eg: Realization of AND, OR that are linear seperable.

⑨ Example:

Hebb Network, Hopfield Network

⑥. Adv of MLNN:

→ Training a MLNN is difficult because of the existense of hidden units.

→ Output depends upon the interconnections information (weights) assd inputs presented in hidden layer and input units.

⑦. They can solve more complicated problems which can't be solved by single layer Networks.

→ It can perform pattern classification with more than two classes.

⑧ It can be used for classification of pattern that are linear or non-linear seperable.

Eg: Realization of XOR fn is non-linear seperable.

⑨ Example

Back propagation Net,
Counter propagation Net.

(F) Different Types Of ANNs:

→ The arrangement of neurons into layers and pattern of connection between layers are called as architecture of network.

→ The neurons with in a layer are either found to be fully interconnected or not fully connected.

→ The no of layers of neurons are either one or many.

→ Layers are connected to each other by weighted interconnections.

→ Thus, depending upon the type of interconnections and no of layers, there are a no of different possible architectures.

(1) Feed forward Network:

→ In this architecture, the activations of the input units are set and then propagated throughout the network until the values of the output units are determined.

→ It is a simple network consisting of an input layer, an output layer and one or more layers of neurons.

→ The main advantage of this network is that it learns to evaluate and recognize input patterns.

→ The network acts as a vector-valued function taking one vector on the input and returning another vector on output.

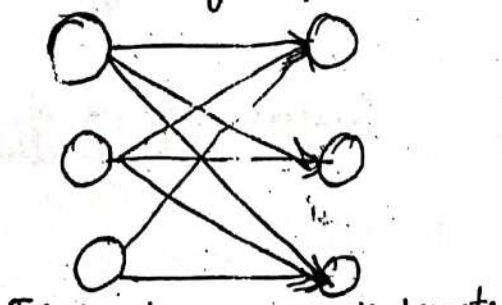
→ They are further classified into two categories:

i) Single layer feed forward :- (SLFF)

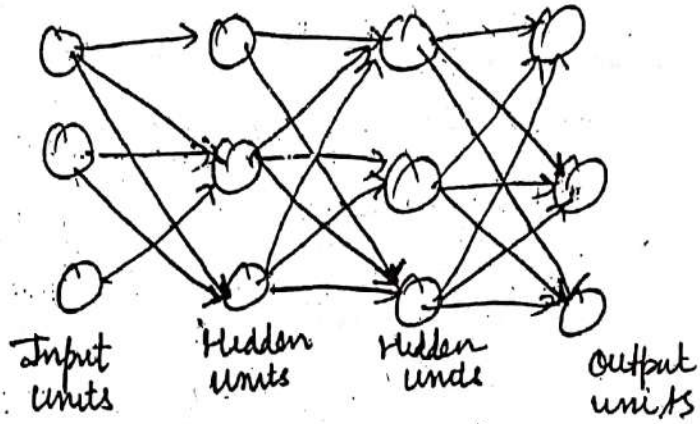
→ It has only one layer of weighted interconnections.

→ The input layer may or may not be fully connected to output units.

→ The weights from one output unit does not influence the weights for other output units



(Single layer FF)



(Multi Layer FF)

ii) Multi Layer feed forward :- (MLFF)

→ It consist of multiple layer of neurons between the input units and output units. These are called as Hidden layer.

→ Signal flows from input units to output units in a forward direction.

→ They are advantageous as compared to single layer neural network as they can solve more complex problems which can't be solved by single layer networks.

→ Example of feed forward Network are Perceptron & Adaline. Back Propagation Network

(2) Feedback ANN :- (Recurrent Network).

→ In these types of ANN, the output goes back into the network to achieve the best-evolved results internally.

→ The feedback network feeds information back into itself and is well suited for optimization problems.

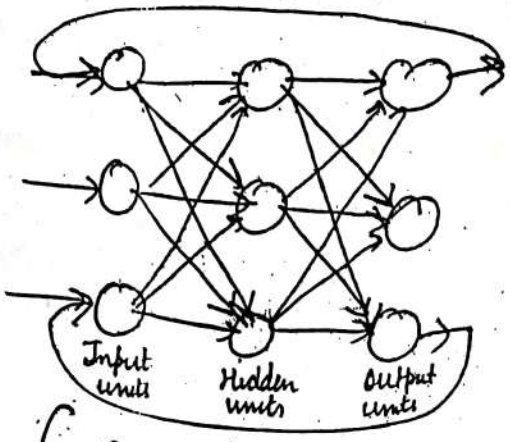
→ They are used by internal system error connections

→ They also allow networks to process sequential information.

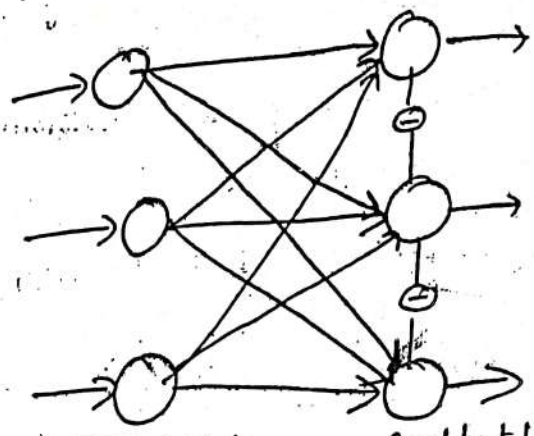
→ Processing in recurrent network depends on the state of network at the last time step.

→ Moreover, the response to the current input depends on previous inputs.

→ Examples: Kohonan & Hopfield network.



(Recurrent Network)



(Competitive Network).

(3) Competitive ANNs :-

→ It is similar to SLFF network except that there are connections usually negative, between the output nodes. Because of these connections the output nodes tend to compete to represent input pattern.

→ Sometimes the output layer is completely connected and sometimes the connections are restricted that are closer to each other.

→ Networks of this kind have been used to explain formation of topological (maps)
(shows vital information and removes unnecessary details.

→ Competitive learning is a form of unsupervised learning in ANN, in which nodes compete for the right to respond to a subset of input data.

TRAINING OF ARTIFICIAL NEURAL NETWORKS :

→ The method of setting the value of weights enables the process of training.

→ Training a network involves modifying the weights in the interconnection between layers so as to achieve the expected output.

→ The internal process that takes places when a network is trained is called learning.

→ Generally 3 types of training are as follows:

1) Supervised Training : (with a 'Teacher')

→ It is the process of providing the network with a series of sample inputs and comparing the outputs with expected responses.

→ The training continues until the network is able to provide expected responses.

→ The weights are adjusted according to a learning algorithm.

→ Examples: Hebb net, Back propagation network.

2) Unsupervised Training: (without a 'Teacher')

→ In a neural net, if for the training input vectors, the target output is not known, then the training method adopted is called as unsupervised Training.

→ The net may modify the weight so that the most similar input vector is assigned to the same output unit.

→ Unsupervised networks are complex and difficult to implement.

→ Unsupervised networks are also called as self-organising network or self learning network because of their ability to carry out self learning (as network involves looping connections back into feedback layer and iterating until some sort of stable recall can be achieved)

→ Examples: Self-Organising maps, Adaptive Resonance Theory (ART)

3) Reinforcement Training:

→ In this method, a teacher is assumed to be present but right answer is not presented to the network.

→ The network is only presented with an indication of whether the output answer is right or wrong

→ This type of training is applied if supervised learning is not available.

→ The output in this case may not be indicated as desired output, but the condition whether it is success (1) or failure (0) can be indicated.

→ Based on this error can be calculated and network using hit & trial error for input - output mapping so as to maximize performance.

Learning Rules :

→ A neural network learn about its environment through an interactive process of adjustments applied to its synaptic weights and bias levels.

→ The set of well defined rules for the solution of a learning problem is called a learning algorithm.

→ Each learning algo differs from the other in the way in which the adjustment to a synaptic weight of a neuron is formulated.

1) Hebbian Learning Rule :

→ It is the oldest and most famous of all learning rules.

→ It states that "when an axon of cell A is near enough to excite a cell B and repeatedly takes part in firing it, some metabolic changes or growth process takes place in one or both cells such that A's efficiency as one of the cells firing B, is increased". This can also be called as correlational learning.

→ This statement may be split into two-part rule:

i) If two neurons on either side of a synapse are activated simultaneously, then the strength of that synapse is selectively increased.

ii) If two neurons on either side of a synapse are activated asynchronously, then that synapse is selectively weakened or eliminated.

→ The simplest form of Hebbian learning is described by:

ΔW = change in weight

$$\Delta W = x_i y$$

x_i = set of input
y = output

→ This learning rules represents a purely feed forward, unsupervised learning.

→ It states that if the cross product of output and input is positive, this results in increase of weight, otherwise the weight decreases.

2) Perceptron Learning Rule:

→ Basic Concept: As being supervised in nature, to calculate the error, there would be a comparison between the desired / target output and the actual output. It is used in supervised learning of neural network.

→ If there is any difference found between target output and actual output then a change must be made to the weights of connection.

→ Mathematically,

Let suppose we have 'n' no of finite input vectors i.e $x(n)$

And we have associated target output vectors $t(n)$ where $n = 1$ to N

Output 'y' can be calculated on the basis of activation function being applied over net input.

It can be expressed as.

$$y = \begin{cases} 1 & \text{if } y > \theta \\ 0 & \text{if } -\theta \le y_{in} \le \theta \\ -1 & \text{if } y_{in} < -\theta \end{cases}$$

θ = Threshold

→ The updating of weight can be done in the following two cases:

Case-I: when $t \neq y$ then

$$w(new) = w(old) + tx$$

t = target output
y = Actual output

Case-II: when $t = y$, then

No change in weight

3) Delta Learning Rule (Widrow-Hoff Rule or Least Mean Square (LMS) Rule)

→ It is introduced by Bernard Widrow and Marcian Hoff, used to minimize the error over all training patterns. (main aim of delta rule)

→ It is a kind of supervised learning algorithm with having continuous activation functions.

→ Delta Rule may be stated as:

"The adjustment made to a synaptic weight of a neuron is proportional to the product of the error signal and the input signal of the synapse.

Mathematically, $\boxed{\Delta W = \alpha . (t - y_{in}) . x_i}$ ← only for single output unit

ΔW = change in weight

α = learning rate

$(t - y_{in})$ = difference between the target output & actual output.

x_i = input value.

→ Delta rule can be applied for single output unit and several output units.

→ Updating of weight can be done in following two cases:

Case-I: when $t \neq y$ then

$$w(new) = w(old) + \Delta w$$

Case-II: when $t = y$ then

No change in weight.

$\boxed{\Delta w = \alpha . (t_j - y_{inj}) . x_i}$ ← This is weight adjustment for several output unit

y_{inj} = output from the Ith input unit to the Jth output unit.

t_j = target of J^{th} output unit

4) Competitive Learning Rule:

→ There will be a competition among the output nodes

→ The main concept is that during training, the output unit with the highest activation to a given input pattern will be declared the winner

→ This rule is also called as Winner takes all.

Mathematically, 3 important factors are there for this rule.

a) Condition to be winner: Suppose if a neuron N wants to be winner then there would be following condition:

$$N = \begin{cases} 1 & \text{if } V_K > V_j \text{ for all } j \quad j \neq K \\ 0 & \text{otherwise} \end{cases}$$

It means if any Neuron, say N, wants to win, then its induced local field (the output summation unit), say V_K must be largest among all the other neurons in the network.

b) Condition of sum of total of weight:
The sum total of weights to a particular output neuron should be 1.

For example, if we consider neuron K, then

$$\sum_j w_{Kj} = 1 \quad \text{for all } K.$$

c) Change of weight for winner:

→ If a neuron does not respond to the input pattern, then no learning takes place in that neuron.

→ However if a particular neuron wins, then the corresponding weights are adjusted as:

$$\Delta w_{kj} = \begin{cases} -\alpha (x_j - w_{kj}), & \text{if neuron } k \text{ wins} \\ 0, & \text{if neuron } k \text{ loses} \end{cases}$$

α = learning rate.

→ This rule is suited for unsupervised network training.

5) Out star learning rule:

→ This rule, introduced by Grossberg, is concerned with supervised learning because desired outputs are known. It is also called Grossberg learning.

→ Basic Concept: This rule is applied over the neurons arranged in a layer. It is specially designed to produce a desired output d of the layer of p neurons.

Mathematically, weight adjustment are computed as

$$\Delta w_j = \alpha (d - w_j)$$

d = desired neuron output

α = learning rate.

6) Boltzman Learning :

→ In this learning, the neurons constitute a recurrent structure and they work in binary form

→ This learning is characterized by an energy function E, given by.

$$E = -\frac{1}{2} \sum_i \sum_j w_{ij} x_j x_i \quad [i \neq j$$

It is stochastic learning because neurons are selected random for operation

x_i is the state of neuron i

w_{ij} is the weight from neuron i to neuron j

$i \neq j$ means none of the neuron in the machine has self-feedback.

→ Neurons of this learning process are divided into two groups

- Visible (There is an interface between the network and the environment in which it operates).
- Hidden (They operates independent of the environment)

7) Memory-Based Learning :

→ In this, all the previous experiences are stored in a large memory of correctly classified input-output examples :

$$(x_i, t_j)_{i=1}^{N}$$ where, x_i = input vector, t_j = desired response.

→ The memory based algorithm involves two facts:

a) criterion used for defining the local neighborhood of the test vector.

b) learning rule applied to the training in the local neighbourhood.

Eg: Nearest neighbour rule.

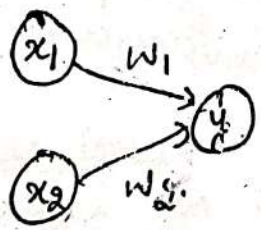
(I) ARTIFICIAL NEURAL NETWORK (ANN) TERMINOLOGIES

1) Weights:

→ A neural network consist of large no of neurons.

→ These neurons are connected to each other by directed communication links, which are associated with weights.

→ Weight is an information used by the neural network to solve a problem.



x_1 = Activation of neuron 1
x_2 = Activation of neuron 2
y = Output neuron
w_1 = Weight connecting neuron 1 to output
w_2 = Weight connecting neuron 2 to output

→ Based on all parameters, the net input 'Net' can be calculated as the summation of the product of weights and input signals.

$$Net = x_1 w_1 + x_2 w_2$$

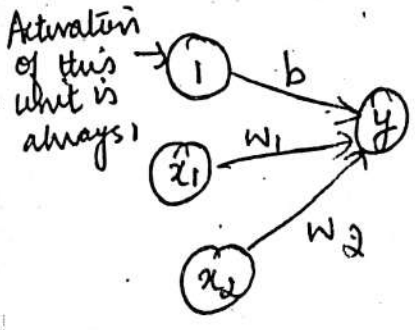
$$Net = \sum_i x_i w_i$$

2) Bias:

→ A bias acts exactly as a weight on a connection from a unit whose activation is always 1.

→ Increasing the bias, increases the net input

→ The bias improves the performance of the neural network.



If bias is present, the net input is calculated as:

$$\text{Net} = b + \sum x_i w_i$$

3) Threshold:

→ The threshold 'θ' is a factor which is used in calculating the activations of the given net.

→ Based on the threshold, output can be calculated i.e activation function is based on the value of θ:

Eg: $$y = f(\text{Net}) = \begin{cases} +1 & \text{if net} \geq \theta \\ -1 & \text{if net} < \theta \end{cases}$$

4) Activation Functions:

→ The activation function is used to calculate the output of a ~~response~~ neuron.

→ The sum of the weighted input signal is applied with an activation to obtain response.

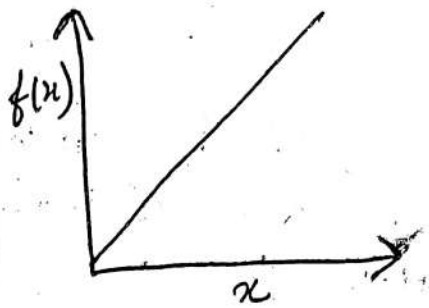
→ They may be linear or non linear.
→ The non linear activation fns are used in multi layer net.

a) Identity function

The function is given by,

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

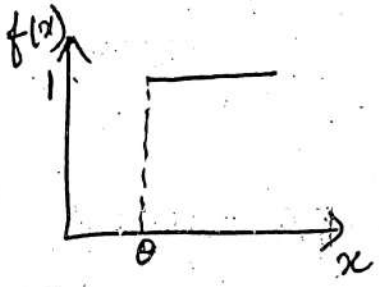
(Can't applied to multiLayer NNs).



b) Binary Step Function

→ Also called as Heaviside fn or Threshold fn.

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



c) Sigmoidal functions:

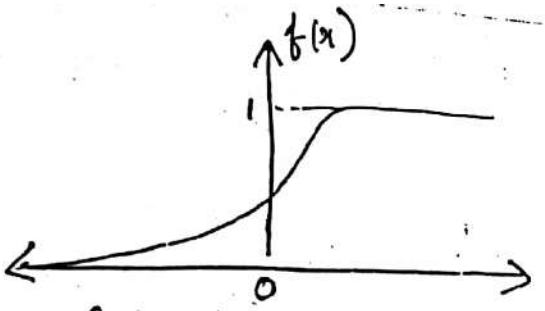
→ These functions are usually S-shaped curves
→ The hyperbolic and logistic functions are commonly used.
→ They are used in multi layer networks like Back propagation network, Radial basis function.
→ There are two main types of sigmoidal functions

i) Binary Sigmoidal fn:

→ Also called logistic function.
→ Ranges between 0 to 1.

$$f(x) = \frac{1}{1 + \exp(-\sigma x)}$$

σ = Steepness parameter.

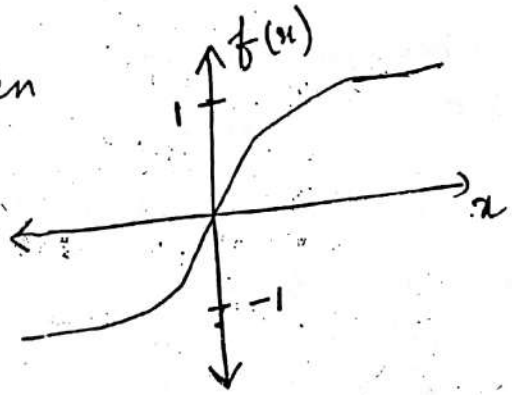


(Binary sigmoidal function)

ii) Bipolar Sigmoidal fn:

→ The desired range is between +1 and -1

→ $$f(x) = \frac{1 - \exp(-\sigma x)}{1 + \exp(-\sigma x)}$$



Advantages:

→ Less computational burden during training.

Disadvantages:

→ It can be used only in some specific multi-layer networks.

→ Involves the use of exponent 'e' factor

(J) LINEAR SEPARABLE CONCEPT (Linear separability)

→ Linear separable means that there is a line which seperates points of one class from other class.

→ In general, for any output unit, the desired response is '1' if corresponding input is a member of class or '0' if it is not.

→ The activation fn is taken as step fn. This fn returns a high 1 if net input is positive and a low 1 if net input is negative.

→ The relation, $\boxed{y_{in} = b + \sum_i x_i w_i = 0}$ gives the boundary region of the net input.

→ The boundary between the region where $y_{in} > 0$ and $y_{in} < 0$ is called 'decision boundary'.

→ The equation denoting this decision boundary can represent a line, plane.

→ On training, if the weights of training input vectors of correct response +1 lie on one side of the boundary and of the training input vectors of response −1 lie on other side of the boundary, then the problem is <u>linear seperable</u> else it is linearly non-seperable

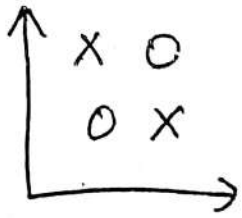
→ Say, with two input vectors, the equation of the line separating the positive region and negative region is given by,

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

$$\boxed{x_2 = \frac{-b}{w_2} - x_1 \frac{w_1}{w_2}}$$

These two regions are called the decision region of the net.

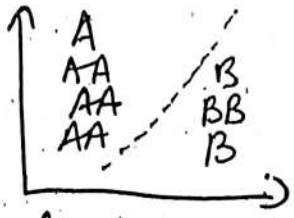
Look at following data sets:



(Not linear separable)

↓

(You can't draw straight line so that all X are on one side & O on other side).



(Linear separable)

(K) PERCEPTRONS :

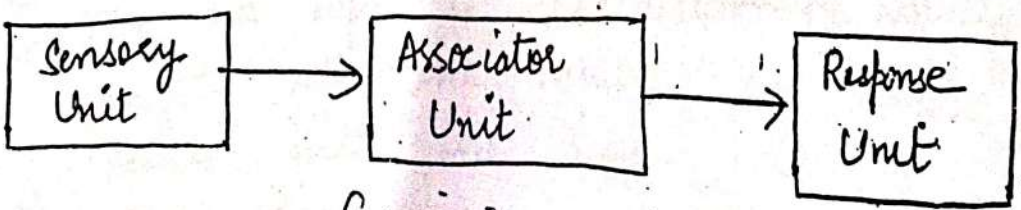
→ Rosenblatt, Minsky & Papert developed large class of ANN called as perceptrons.

→ It employs supervised learning rule and is able to classify the data into two classes.

→ The perceptrons use threshold output function and the McCulloch-Pitts model of a neuron.

→ Their iterative learning converges to correct weights.

→ The original perceptron is found to have 3 layers sensory, associator and response unit.



(Original perceptron)

→ The sensory and association units have binary activations and an activation of +1, 0, -1 is used for the response unit.

→ The perceptron are of two types:

- Single layer Perceptron
- Multi layer Perceptron.

⇒ Single layer Perceptron :- (SLP)

→ It is the simplest form of a neural network used for the classification of patterns that are linear seperable.

→ Basic Concept: It consist of single neuron with adjustable weights and bias.

→ It says if the patterns used to train the perceptron are drawn from two linearly seperable classes, the perceptron algorithm converges and positions the decision surface in the form of a hyperplane between the two classes."

→ The perceptron built around a single neuron is limited to performing pattern classification with only two classes.

→ Training in perceptron continues till no error occurs.

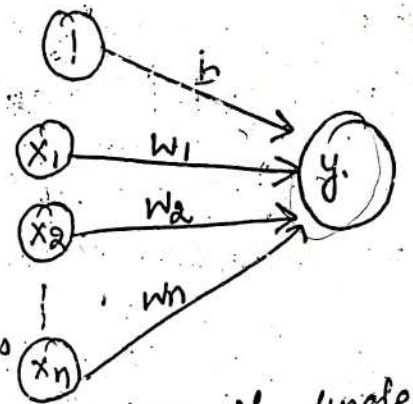
⇒ Architecture of single layer perception.

→ As discussed earlier perception has sensory, associator and response unit. But in figure only associator unit and response unit is shown, because weights between associator & response units are adjusted.

→ The input layer consist of input neurons from X_1 —— X_n.

→ There always exists a common bias of '1'.

→ The input neurons are connected to output neurons through weighted interconnections



(Architecture of single layer perception)

→ Perception network can be trained for single output unit as well as multiple output units.

⇒ Training Algorithm :

a) To start the training process, initially the weights & bias are set to zero.

b) α (learning Rate parameter) is set which ranges between 0 to 1.

c) The net input is calculated by multiplying the weights with the inputs and adding the result with the bias.

d) Once the net input is calculated, ~~threshold~~ activation

function is applied, to calculate the output of the network.

e) This output is compared with the target, if any difference occurs we go for weight updation based on perceptron learning rule else the network training is stopped.

f) Algorithm can be used for binary and bipolar input vectors.

g) It uses a bipolar target with fixed threshold and adjustable bias.

Training Algorithm for Single Output Unit:

Step-I: Initialize weights and bias (initially to zero)
Set learning rate α (0 to 1).

Step-II: While stopping condition is false do steps 3-7.

Step-III: For each training pair s:t do steps 4-6.

Step-IV: Set activations of input units.
$x_i = s_i$ for $i = 1$ to n.

Step-V: Compute the output unit response:

$$y_{in} = b + \sum_i x_i w_i.$$

The activation function used is

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

* If $y_j \neq t_j$ and $x_i \neq 0$ then

$$w_{ij}(new) = w_{ij}(old) + \alpha t_j x_i$$
$$b_j(new) = b_j(old) + \alpha t_j$$

else if $y_j = t_j$

$$w_{ij}(new) = w_{ij}(old)$$
$$b_j(new) = b_j(old)$$

⇒ Multi layer Perceptron :- (MLP)

→ MLP are used with supervised learning and have led to the successful back propagation algo.

→ The disadvantage of SLP is that they can't be extended to multi layered version.

→ MLP uses non-linear activation function.

→ MLP network also has various layer of hidden neurons.

→ Hidden neurons make MLP network active for highly complex task.

Disadvantage of MLP : 1) Presence of non-linearity and complex connections of the network leads to highly complex theoretical analysis

2) Existence of hidden neurons makes learning process tedious.

→ MLPs are multi layer fully connected networks.

→ Examples: Back propagation Network, Radial basis function network.

Step - VI: The weights and bias are updated if the target is not equal to output response.

If $t \neq y$ and the value of x_i is not zero

$$w_{ij(new)} = w_{ij(old)} + \alpha t_j x_{ij}$$
$$b(new) = b(old) + \alpha t_j$$

else $t_j = y_j$

$$w_{ij(new)} = w_{ij(old)}$$
$$b(new) = bold.$$

Step- VII: Test for stopping condition, which would happen when there is no change is weight

⇒ Training Algorithm for several Output Classes:

(whole Algorithm is same as above)
only the output units are increased

(Use Above Algo with following modifications)

Activation output of each output unit $y_{inj} = b_j + \sum x_i w_i$ for $j = 1$ to m

Step-5: $y_j = f(y_{inj}) = \begin{cases} 1 & \text{if } y_{inj} > \theta \\ 0 & \text{if } -\theta \leq y_{inj} \leq \theta \\ -1 & \text{if } y_{inj} < -\theta \end{cases}$

Step-6: Weights and bias are to be updated for $j = 1$ to m and $i = 1$ to n.

CHAPTER 3

Fundamental Models of Artificial Neural Networks

What You Will Learn?

- The basic fundamental neuron model, i.e. the McCulloch-Pitts neuron model.
- Excitatory and inhibitory connection paths in the McCulloch-Pitts neuron.
- Process of learning in a neural network.
- Types of learning algorithms and their classification based on the way in which the adjustment to a synaptic weight of a neuron is formulated.
- Various learning algorithms like Hebb learning rule, perceptron learning rule, Delta rule, etc.
- The linear separability concept used to form the decision boundary regions.

3.1 Introduction

The development of the neural network started with the introduction of McCulloch-Pitts neuron. The learning Law was introduced by Hebb, which resulted in Hebb net. In this chapter, the fundamental models of Artificial Neural Network (ANN) are dealt along with the various learning rules and linear separability concepts.

3.2 McCulloch-Pitts Neuron Model

The first formal definition of a synthetic neuron model based on the highly simplified considerations of the biological model was formulated by Warren McCulloch and Walter Pitts in 1943. The McCulloch-Pitts model of a neuron is characterized by its formalism, elegant, and precise mathematical definition.

McCulloch-Pitts neuron allows binary 0 or 1 states only, i.e. it is binary activated. These neurons are connected by direct weighted path. The connected path can be excitatory or inhibitory. Excitatory connections have positive weights and inhibitory connections have negative weights. There will be same weights for the excitatory connection entering into a particular neuron. The neuron is associated with the threshold value. The neuron fires if the net input to the neuron is greater than the threshold. The threshold is set so that the inhibition is absolute, because, non-zero inhibitory input will prevent the neuron from firing. It takes only one time step for a signal to pass over one connection link.

3.2.1 Architecture

The architecture of the McCulloch Pitts neuron is shown in Fig. 3.1.

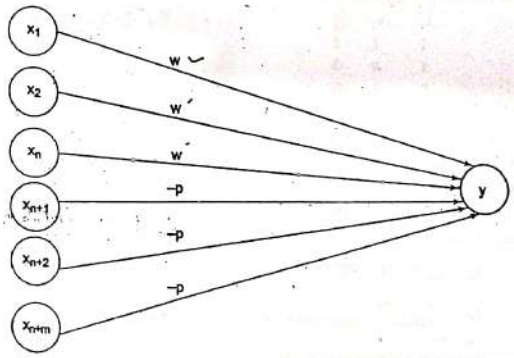


Fig. 3.1 Architecture of a McCulloch-Pitts Neuron

'Y' is the McCulloch-Pitts neuron, it can receive signal from any number of other neurons. The connection weights from $x_1, \ldots x_n$ are excitatory, denoted by 'w' and the connection weights from

$x_{n+1} \cdots x_{n+m}$ are inhibitory denoted by '– p'. The McCulloch-Pitts neuron Y has the activation function,

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

where θ is the threshold and y_{in} is the total net input signal received by neuron Y.

The threshold θ should satisfy the relation.

$$\theta > nw - p$$

This is the condition for absolute inhibition.

The McCulloch-Pitts neuron will fire if it receives k or more excitatory inputs and no inhibitory inputs, where

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

Solved Examples

Example 3.1 Generate the output of logic AND function by McCulloch-Pitts neuron model.

Solution The AND function returns a true value only if both the inputs are true, else it returns a false value. '1' represents true value and '0' represents false value.

The truth table for AND function is,

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

A McCulloch-Pitts neuron to implement AND function is shown in Fig. 3.2. The threshold on unit Y is 2.

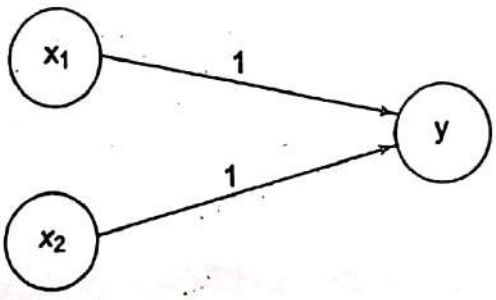


Fig. 3.2 McCulloch-Pitts Neuron to Perform Logical AND Function

The output Y is,

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

The net input is given by

$$y_{in} = \sum_i \text{weights} * \text{input}$$

$$y_{in} = 1 * x_1 + 1 * x_2$$

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

From this the activations of output neuron can be formed.

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

Now present the inputs

(i) $x_1 = x_2 = 1$, $y_{in} = x_1 + x_2 = 1 + 1 = 2$

$y = f(y_{in}) = 1$ since $y_{in} = 2$.

(ii) $x_1 = 1, x_2 = 0, y_{in} = x_1 + x_2 = 0 + 1 = 1$

$y = f(y_{in}) = 0$ since $y_{in} = 1 < 2$.

This is same when $x_1 = 0$ and $x_2 = 1$.

(iii) $x_1 = 0$, $x_2 = 0$, $y_{in} = x_1 + x_2 = 0 + 0 = 0$.

Hence, $y = f(y_{in}) = 0$ since $y_{in} = 0 < 2$.

Example 3.2 Generate OR function using McCulloch-Pitts neuron model.

Solution The OR function returns a high ('1') if any one of the input is high, returns a low ('0') if none of the inputs is high.

The truth table for OR function is,

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

A McCulloch-Pitts neuron for OR functions is shown in Fig. 3.3. The threshold for the unit is 3.

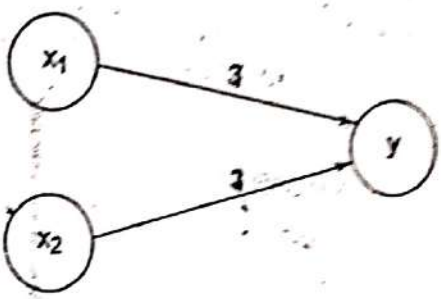


Fig. 3.3 McCulloch-Pitts Neuron for OR Function

The net input is calculated as,

$$y_{in} = 3x_1 + 3x_2$$

The output is given by,

$$y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} \geq 3 \\ 0 & \text{if } y_{in} < 3 \end{cases}$$

Presenting the inputs,

(i) $x_1 = x_2 = 1$, $y_{in} = 3x_1 + 3x_2 = 1 + 1 = 2$

$= 3 \times 1 + 3 \times 1 = 6 >$ threshold 3.

Hence, $y = 1$

(ii) $x_1 = 1$, $x_2 = 0$,

$y_{in} = 3x_1 + 3x_2$

$= 3 \times 1 + 3 \times 0 = 3 =$ threshold

Applying activation formula

$y = f(y_{in}) = 1$

This is also the case when $x_1 = 0$ and $x_2 = 1$

(iii) $x_1 = x_2 = 0$,

$y_{in} = 3x_1 + 3x_2 = 3 \times 0 + 3 \times 0 = 0 <$ threshold

Hence output $y = 0$.

Example 3.3 Realize NOT function using McCulloch-Pitts neuron model.

Solution The NOT function returns a true value ('1') if the input is false ('0') and returns a false value ('0') if the input is true ('1').

The truth table for NOT function is,

x	y
1	0
0	1

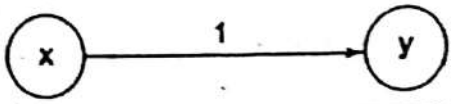


Fig. 3.4 McCulloch-Pitts Neuron for NOT Function

The McCulloch-Pitts neuron for this function is given in Fig. 3.4. The threshold for unit y is 1.

The net input is,

$$y_{in} = x \cdot w$$

Since $w = 1$, $y_{in} = x$.

The output activation is given by,

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

Presenting the input,

(i) $x_1 = 1$, $y_{in} = 1$

Applying activation,

$$y = f(y_{in}) = 0$$

(ii) $x_1 = 0$, $y_{in} = 0$

Applying activation

$$y = f(y_{in}) = 1.$$

Example 3.4 Generate the output of ANDNOT function using McCulloch-Pitts neuron.

Solution The ANDNOT function returns a true value ('1') if the first input value is true ('1') and the second input value is false ('0').

The truth table for the ANDNOT functions is,

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

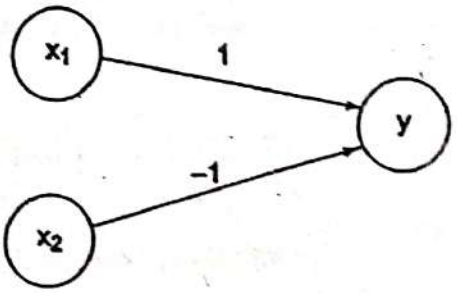


Fig. 3.5 McCulloch-Pitts Neuron for ANDNOT Function

The McCulloch-Pitts neuron for the ANDNOT function is shown in Fig. 3.5. The threshold of unit Y is 1.

The net input is,

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

$$= x_1 * 1 + x_2 * -1$$

$$y_{in} = x_1 - x_2$$

The output activation is given as,

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

Presenting the input

(i) $x_1 = x_2 = 1$, $y_{in} = x_1 - x_2 = 1 - 1 = 0 < 1$

Hence, $y = f(y_{in}) = 0$

(ii) $x_1 = 1, x_2 = 0$, $y_{in} = x_1 - x_2 = 1 - 0 = 1 = 1$

$y = f(y_{in}) = 1$

(iii) $x_1 = 0, x_2 = 1$, $y_{in} = x_1 - x_2 = 0 - 1 = -1 < 1$

$y = f(y_{in}) = 0$

(iv) $x_1 = x_2 = 0$, $y_{in} = x_1 - x_2 = 0 - 0 = 0 < 1$

$y = f(y_{in}) = 0$.

Thus AND NOT function is realized.

Example 3.5 Realize the Exclusive-OR function using McCulloch-Pitts neuron.

Solution XOR function returns a true value if exactly one of the input values is true; otherwise it returns the response as false. The truth table for XOR function is,

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

The McCulloch-Pitts neuron model for this is given in Fig 3.6. The threshold of unit y is 1.

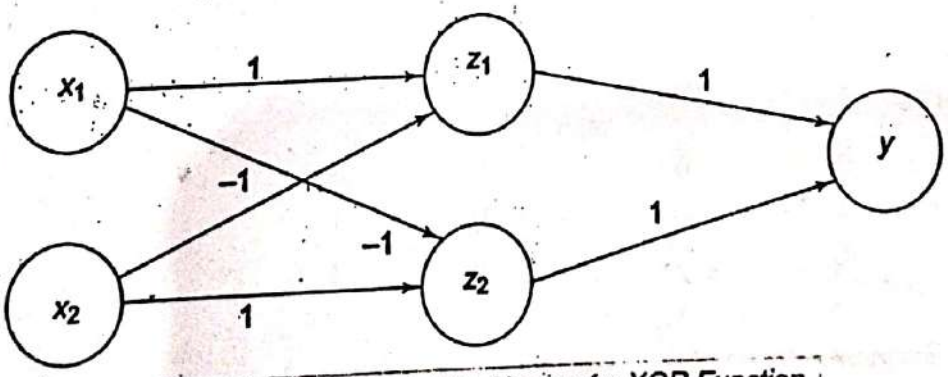


Fig 3.6 McCulloch-Pitts Neuron for XOR Function

With one layer alone, it was not able to predict the value of the threshold for the neuron to fire, hence another layer is introduced.

$$x_1 \text{ XOR } x_2 = (x_1 \text{ ANDNOT } x_2) \text{ OR } (x_2 \text{ ANDNOT } x_1)$$

$$x_1 \text{ XOR } x_2 = z_1 \text{ OR } z_2$$

where, $z_1 = x_1$ ANDNOT x_2

and $z_2 = x_2$ ANDNOT x_1

The activations of z_1 and z_2 are given as,

$$z_1 = (z_{in-1}) = \begin{cases} 1 & \text{if } z_{in-1} \geq 1 \\ 0 & \text{if } z_{in-1} < 1 \end{cases}$$

$$z_2 = (z_{in-2}) = \begin{cases} 1 & \text{if } z_{in-2} \geq 1 \\ 0 & \text{if } z_{in-2} < 1 \end{cases}$$

The calculation of net input and activations of z_1 and z_2 are shown below.

$z_1 = (x_1 \text{ ANDNOT } x_2)$ $\quad z_{in-1} = x_1 w_1 + x_2 w_2$

x_1	x_2	z_{in-1}	z_1
		$w_1 = 1, w_2 = -1$	
1	1	0	0
1	0	1	1
0	1	−1	0
0	0	0	0

$z_2 = (x_2 \text{ ANDNOT } x_1)$ $\quad z_{in-2} = x_1 w_1 + x_2 w_2$

x_1	x_2	Z_{in-2}	z_2
		$w_1 = -1, w_2 = 1$	
1	1	0	0
1	0	−1	0
0	1	1	1
0	0	0	0

The activation for the output unit y is 1.

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

Presenting the input patterns (z_1 and z_2) and calculating net input and activations gives output of XOR.

Here, $y_{in} = z_1 w_1 + z_2 w_2$

z_1	z_2	y_{in}	$y = z_1$ or z_2
		$w_1 = 1, w_2 = 1$	
0	0	0	0
1	0	1	1
0	1	1	1
0	0	0	0

Thus the Exclusive-OR function is realized.

Hebb Net

The first learning law for artificial neural network was designed by Donald Hebb in 1949. The law states that if two neurons are activated simultaneously, then the strength of the connection between them should be increased. The Hebbian rule developed by McClelland and Rumelhart 1988 formed the Hebb net. This Hebb net consists of bias which acts exactly as a weight on a connection from a unit whose activation is always 1. If the bias is increased, it increases the net input of the unit. For Hebb net, the input and the output data should be in bipolar form. If it is in binary form, the Hebb net cannot learn, which is an extreme limitation of the Hebb rule for binary data. The Hebb rule is also used for training other nets.

3.4.1 Architecture

The architecture of Hebb net is shown in Fig. 3.8.

The shown Hebb net is a single layer net, which consists of an input layer with many input units and an output layer with only one output unit. This is the basic architecture that performs pattern classification. The bias included for the net is found to be '1', which helps in increasing the net input. This architecture resembles a single layer feed forward network as discussed in Section 2.7 of Chapter 2.

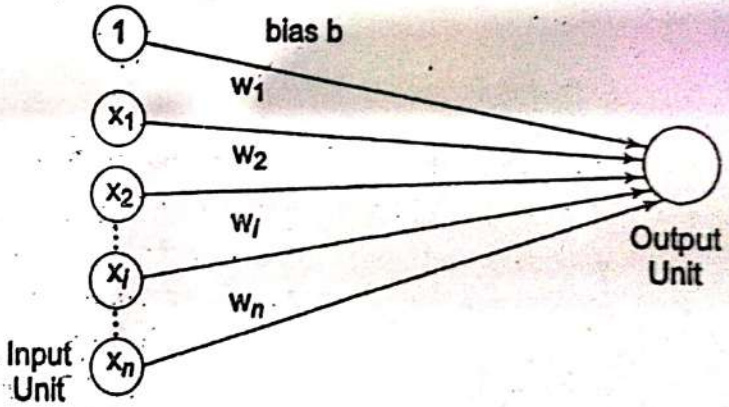


Fig. 3.8 Architecture of a Hebb Net

3.4.2 Algorithm

Initially all the weights and bias are set to zero. Then we can present the input pattern to be classified. At the input layer, the activation function used is identity, hence the output from the input layer remains same as the input presented. Also, the activation for the output unit is also set. Then the weights are updated based on the Hebb learning rule as in Section 3.3.1. An epoch is completed after presenting all the samples of the input pattern. The step wise algorithm to train Hebb net is as follows.

Step 1: Initialize all weights and bias to zero

$w_i = 0$ for $i = 1$ to n. where n is the number of input neurons.

Step 2: For each input training vector and target output pair (S, t) perform Steps 3 – 6.

Step 3: Set activations for input units with input vector.

$x_i = S_i$ ($i = 1$ to n)

Step 4: Set activation for output unit with the output neuron

$y = t$

Step 5: Adjust the weights by applying Hebb rule,

w_i (new) $= w_i$ (old) $+ x_i y$ for $i = 1$ to n.

Step 6: Adjust the bias

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

This algorithm requires only one pass through the training set.

3.4.3 Linear Separability

In general, for any output unit, the desired response is '1' if its corresponding input is a member of class or '0' if it is not. The purpose of training is to make the input pattern to get similar with the training pattern by adjusting the weights.

The activation function is taken as step function. This function retains a high 1 if net input is positive and a low 1 if the net input is negative. The net input to the output neuron is,

$$y_{in} = b + \sum_i x_i w_i$$

The relation,

$$b + \sum_i x_i w_i = 0$$

gives the boundary region of the net input. The boundary between the region where $y_{in} > 0$ and $y_{in} < 0$ is called the 'decision boundary'. The equation denoting this decision boundary can represent a line, plane or hyper plane.

On training, if the weights of training input vectors of correct response +1 lie on one side of the boundary and of the training input vectors of response – 1 lie on the other side of the boundary, then the problem is linear separable else it is linearly non-separable.

Say with two input vectors, the equation of the line separating the positive region and negative region is given by,

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

$$x_2 = \frac{-b}{w_2} - x_1 \frac{w_1}{w_2}$$

These two regions are called the decision regions of the net.

Example 3.9. Realise a Hebb net for the AND function with bipolar inputs and targets.

Solution The AND function gives a high '1' if both the inputs are high else returns a low '–1'.

The training patterns are,

Input			Target
x_1	x_2	B	y
1	1	1	1
1	–1	1	–1
–1	1	1	–1
–1	–1	1	–1

Forming the table, initialize all the weights and the bias to be zero i.e.

$$w_1 = w_2 = 0 \text{ and } b = 0.$$

The weight change is calculated using,

$$\Delta w_i = x_i y \text{ and } \Delta b = y.$$

Input		Target	Weight Changes			Weights		
(x_1	x_2	b)	y	Δw_1	Δw_2	Δb	w_1	w_2	B
						Initial (0	0	0)
1	1	1	1	1	1	1	1	1	1
1	–1	1	–1	–1	1	–1	0	2	0
–1	1	1	–1	1	–1	–1	1	1	–1
–1	–1	1	–1	1	1	–1	2	2	–2

This completes one epoch of training. The straight line separating the regions can be obtained after presenting each input pair. Thus,

$$x_2 = -x_1 \frac{w_1}{w_2} - \frac{b}{w_2}$$

After 1st input,

$$x_2 = -x_1 \frac{1}{1} - \frac{1}{1} = -x_1 - 1$$

$$x_2 = -x_1 - 1$$

Similarly after 2nd, 3rd and 4th epochs, the separating lines are,

$$x_2 = 0, \quad x_2 = -x_1 + 1, \quad x_2 = -x_1 + 1$$

For the 3rd and 4th epoch the separating line remains the same, hence this line separates the boundary regions as shown in Fig. 3.9.

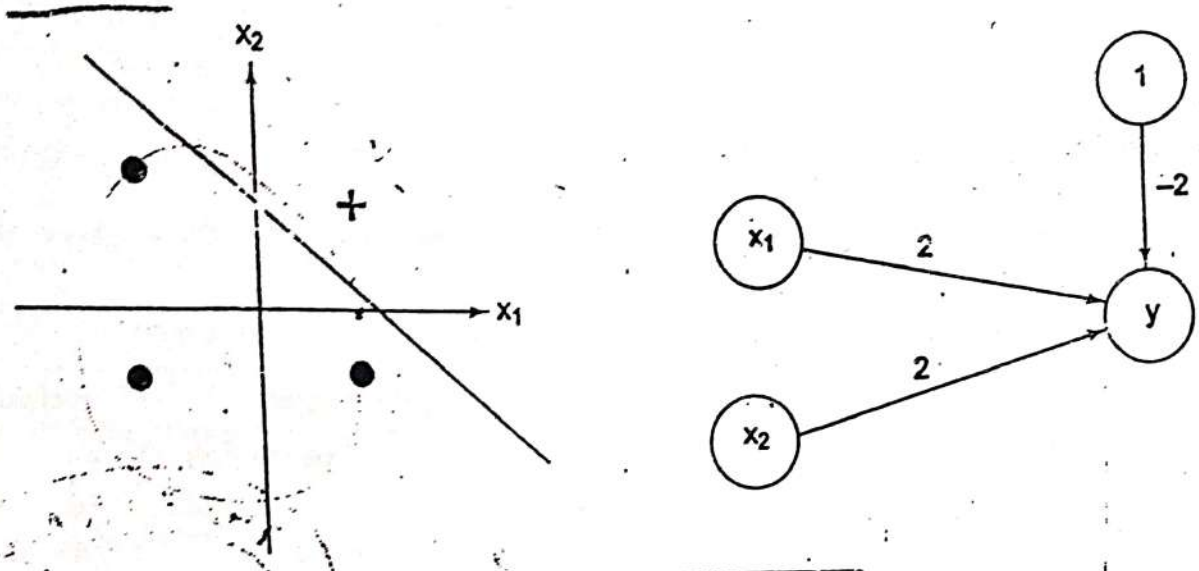


Fig. 3.9 Hebb Net for AND Function

The same procedure can be repeated for generating the logic function OR, NOT, AND NOT etc.

Example 3.10 Apply the Hebb net to the training patterns that define XOR function with bipolar input and targets.

Solution

Input			Target
x_1	x_2	b	y
1	1	1	−1
1	−1	1	1
−1	1	1	1
−1	−1	1	−1

By Hebb training algorithm, assigning initial values of the weights w_1 & w_2 to be zero and bias to be zero.

$w_1 = w_2 = 0$ and $b = 0$

Input			Target	Weight Changes			Weights		
(x_1	x_2	b)	y	Δw_1	Δw_2	Δb	w_1	w_2	B
							(0	0	0)
1	1	1	−1	−1	−1	−1	−1	−1	−1
1	−1	1	1	+1	−1	1	0	−2	0
−1	1	1	1	−1	1	1	−1	−1	1
−1	−1	1	−1	1	1	−1	0	0	0

The weight changes are called using,

$\Delta w_i = x_i y$ and $\Delta b = y$

The new weights by,

$$w_{(n)} = w_{(old)} + \Delta w \text{ and } b_{(n)} = b_{(0)} + \Delta b.$$

The final weights obtained for the XOR function are $w_1 = w_2 = 0$ and $b = 0$. Hence it is clear that the separating line cannot be drawn.

Thus, Hebb rule cannot be used to form a training pattern to define XOR function.