Unit 6 — 9 hrs.

Applications of AI

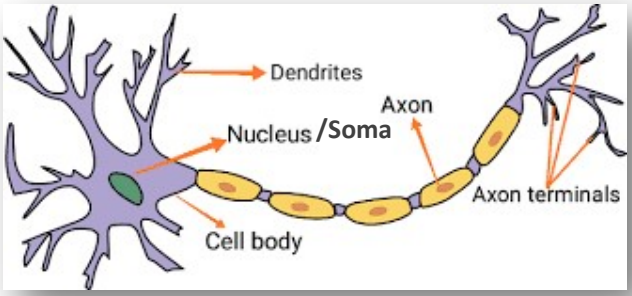
6.1 Neural Network

Biological Neural Network (BNN):

A neural network can be defined as a model of reasoning based on the human brain. The brain consists of a densely interconnected set of nerve cells, or basic information-processing units, called neurons.

The human brain incorporates nearly 10 billion neurons and 60 trillion connections, synapses, between them. By using multiple neurons simultaneously, the brain can perform its functions much faster than the faster computers in existence today.

Each neuron has a very simple structure, but an army of such elements constitutes a tremendous processing power.



Neuron: Fundamental functional unit of all nervous system tissue.

Soma: Cell body, contain nucleus.

Dendrites: A number of fibers, input.

Axon: Single long fiber with many branches, output.

Synapse: Junction of dendrites and axon, each neuron form synapse with 10 to 1,00,000 other neurons.

Biological Neuron	Artificial Neuron
Cell Nucleus (Soma)	Node
Dendrites	Input
Synapse	Weights or interconnections
Axon	Output

How does brain work

Signals are propagated from neuron to neuron by electrochemical reaction:

1. Chemical substances are released from the synapses and enter the dendrite, raising or lowering the electrical potential of the cell body.
2. When a potential reaches a threshold, an electrical pulse or action potential is sent down the axon.
3. The pulse spreads out along the branches of axon, eventually reaching synapses and releasing transmitters into the bodies of other cells.
    a. **Excitatory Synapses:** Excitatory synapses increase its likelihood.
    b. **Inhibitory Synapses:** Inhibitory synapses decrease the likelihood of the firing action potential of a cell.

Artificial Neural Network (ANN):

The inventor of the first neurocomputer, Dr. Robert Hecht-Nielsen, defines a neural network as –

"...a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs."

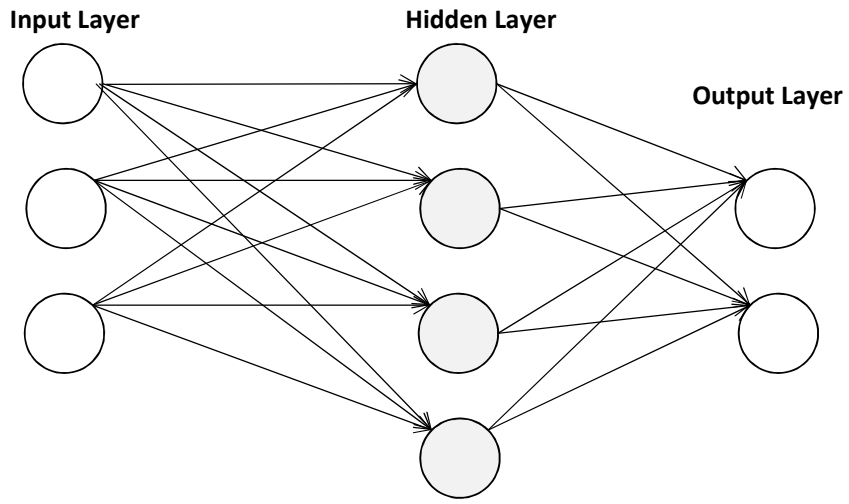
Basic Structure of ANNs

The idea of ANNs is based on the belief that working of human brain by making the right connections, can be imitated using silicon and wires as living neurons and dendrites.

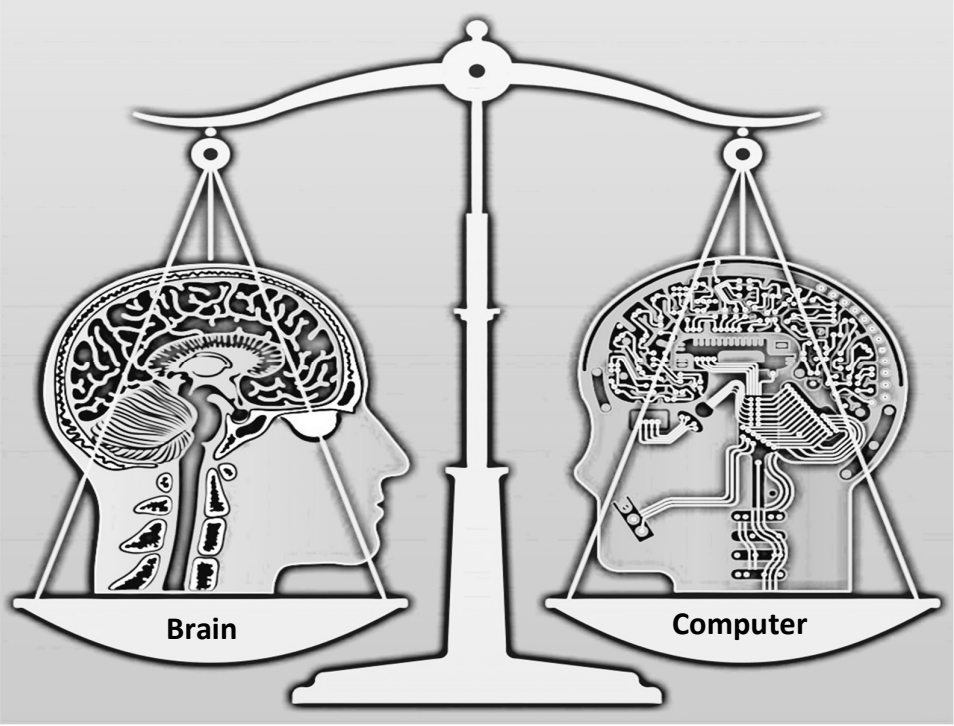
The human brain is composed of 86 billion nerve cells called neurons. They are connected to other thousand cells by Axons. Stimuli from external environment or inputs from sensory organs are accepted by dendrites. These inputs create electric impulses, which quickly travel through the neural network. A neuron can then send the message to other neuron to handle the issue or does not send it forward.

ANNs are composed of multiple nodes, which imitate biological neurons of human brain. The neurons are connected by links and they interact with each other. The nodes can take input data and perform simple operations on the data. The result of these operations is passed to other neurons. The output at each node is called its activation or node value.

Each link is associated with weight. ANNs are capable of learning, which takes place by altering weight values. The following illustration shows a simple ANN –



Brain Vs Computer



Parameters	Brain	Computer
Number of Processor	10^{11}, Neurons	10^{8}, Transistors
Processor complexity	Simple, Inaccurate	Complex, Accurate
Processor Speed	Slow (millisec) 10^{2} Hz	Fast (nanosec) 10^{12} Hz
Style of Computation	Parallel, Distributed	Serial, Centralized
Energy Use	30 W	30 W (CPU)
Learning mode	Learn from experience	Manual programming
Fault Tolerance	Has fault tolerance	No fault tolerance, it gets disrupted when interconnections are disconnected
Storage Capacity	Stores information in its interconnection or in synapse. No loss of memory	Contiguous memory locations. Loss of memory may happen sometimes.

Evolution of ANN:

Year	Neural Network	Designer	Description
1943	McCulloch and Pits Neuron	McCulloch Pitts	Logic Gates
1949	Hebb	Donald Hebb	Strength increases if neurons are active
1958-1988	Perceptron	Rosenblatt	Weights of pat can be adjusted
1960	Adaline	Widrow and Hoff	Mean Squared Error (MSE)
1972	SOM	Kohenen	Clustering
1982	Hopefield	John Hopefield	Associative Memory Nets
1986	Back - Propagation	Rumelhard	Multilayer
1987-1990	ART	Carpenter	Used for both binary and analog

ANN Introduction

An Artificial Neural Network (ANN) is a computational model. It is based on the structure and functions of bilogical neural networks. It works like the way human brain process information.

It includes a large number of connected processing units that work together to process information. They also generate meaningful results from it.

A neural network is an oriented graph. It consists of nodes which in the biological analogy represent neurons, connected by arcs. It corresponds to dendrites and synapses. Each arc associated with a weight at each node.

A neural network is a machine learning algorithm based on the model of a human neuron. The human brain consists of millions of neurons. It spends and process signals in the form of electrical and chemical signals. These neurons are connected with a special structure known as synapses. Synapses allow neurons to pass signals.

An Artificial Neural Network is an information processing technique. It works like the way human brain processes information. ANN includes a large number of connected processing units that work together to process information. They also generate meaningful result from it.

We can apply Neural Network not only for classification, it can also apply for regression of continuous target attributes.

A neural network may contain the following 3 layers:

1. Input layer
2. Hidden layer
3. Output layer

1. **Input layer:** *The activity of the input units represents the raw information that can feed the network.* The purpose of the input layer is to receive as input the value of the explanatory attributes for each observation. Usually, the number of input nodes in an input layer is equal to the number of explanatory variables. Input layer presents the pattern to the network which communicates to one or more hidden layers.

   The nodes of input layer are passive, meaning they do not change the data. They receive a single value on their input and duplicate the value to their many outputs. From the input layer, it duplicates each value and send to all the hidden nodes.

2. **Hidden layer:** *To determine the activity of each hidden unit. The activities of the input units and the weights on the connections between the input and the hidden units. There may be one or more hidden layers.* The hidden layers apply given transformation to he input values inside the network. In this coming arc that go from other hidden nodes or from inputs nodes connected to each node. It connects with going arcs to output nodes or to other hidden nodes.

   In hidden layer, the actual processing is done via a system of weighted connections. There may be one or more hidden layers. The values entering a hidden node multiplied by weights, a set of predetermined numbers stored in the program. The weighted inputs are then added to produce a single number.

3. **Output layer:** *The behavior of the output units depends on the activity of the hidden units and the weights between the hidden and output units.* The hidden layer then link to an output layer. Output layer receives connections from hidden layers or from input layers. It returns an output value that corresponds to the prediction of the response variable. In classification problems, there is usually only one output node. The active nodes of the output layer combine and change the data to produce the output values.

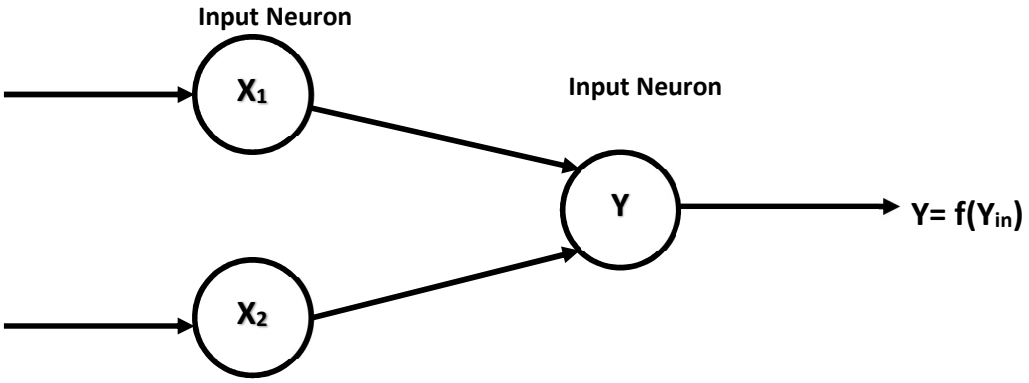
   The ability of the neural network to provide useful data manipulation lies in the proper selection of the weights. This is different from conventional information processing.

6.1.1 Network Structure

Architecture of Artificial Neural – ANN:

- Information-processing system.
- Neurons process the information.
- The signals are transmitted by means of connection links.
- The links possess an associated weight.
- The output signal is obtained by applying activations to the net input.

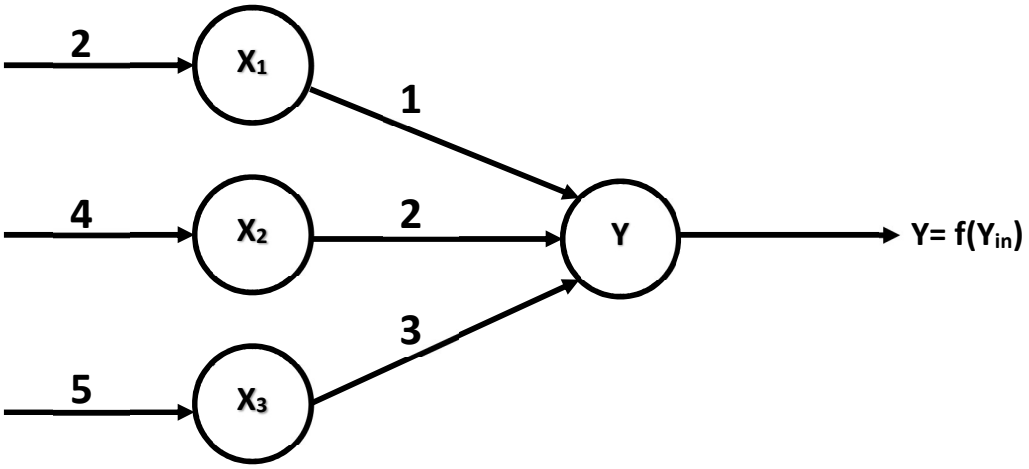
- ✓ ANN causes large number of highly interconnected elements called nods or neurons.
- ✓ Which usually operates in parallel and are configured in regular architecture.
- ✓ Each neuron is connected by other, by a link connection.
- ✓ Each connection is associated with weights which contains information about the input signal.
- ✓ This information is used by a neural network to solve particular problems.
- ✓ Each neuron will have their own internal state.
- ✓ The internal state is called activation or activity level of neuron.
- ✓ The activation can be transferred to one neuron to another neuron.
- ✓ One neuron can send one signal at a time.



- ✓ The function which is applied over the neural net is called "Activation Function".
- ✓ $Y_{in} = X_1W_1+X_2W_2$
- ✓ General Equation of Y_{in}:
- ✓ $Y_{in} = X_1W_1+X_2W_2+\ldots\ldots\ldots\ldots\ldots+X_nW_n =$

$$\sum_{i=1}^{n} X_i * W_i$$

Example:



$Y_{in} = X_1W_1+X_2W_2+X_3W_3$

$Y_{in} = 2*1 + 4*2 + 5*3$

$Y_{in} = 25$ Ans.

Why develop artificial intelligence?

The purpose of building artificial intelligence model of the brain can be neuro-science, they study of the brain and the nervous system in general. It is tempting to think that by mapping the human brain in enough details, we can discover the secrets of human and animal cognition and consciousness.

Basic Model of ANN:

(i) The model's synaptic interconnection
(ii) The training or learning rules adopting for updating and adjusting the connection weights.
(iii) Their activation functions.

The arrangement of neurons to/from layer and the connection patterns formed within and between the layer is called network architecture.

A layer is formed by taking processing element and combining it within other processing element.

An ANN consists of a set of highly interconnected processing elements(neurons) such that each processing elements output is found to be connected through weights to other neurons or itself.

Basically, Neural Nets are classified into:

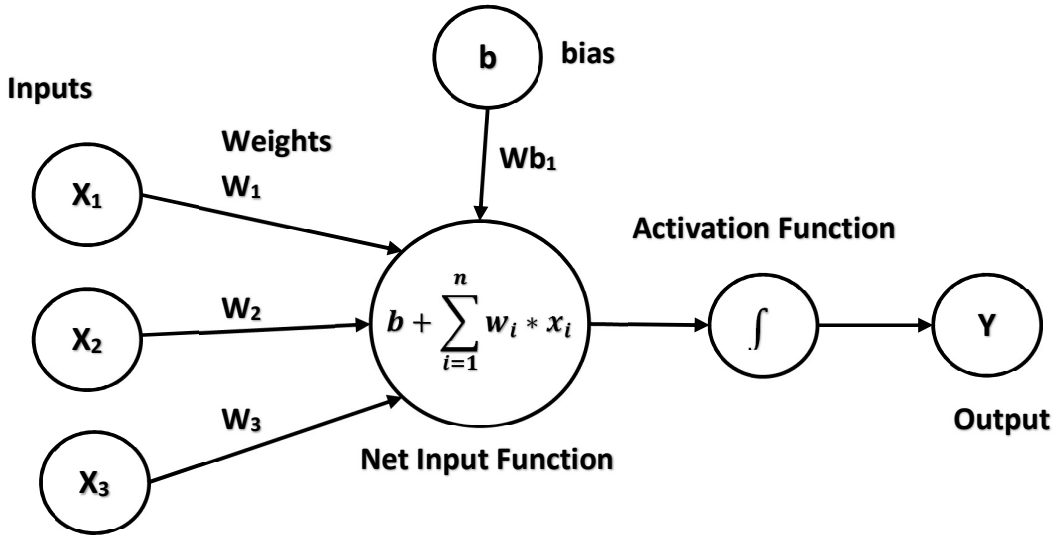
i. Single-layer
ii. Multi-layer

Different types of connections in ANN:

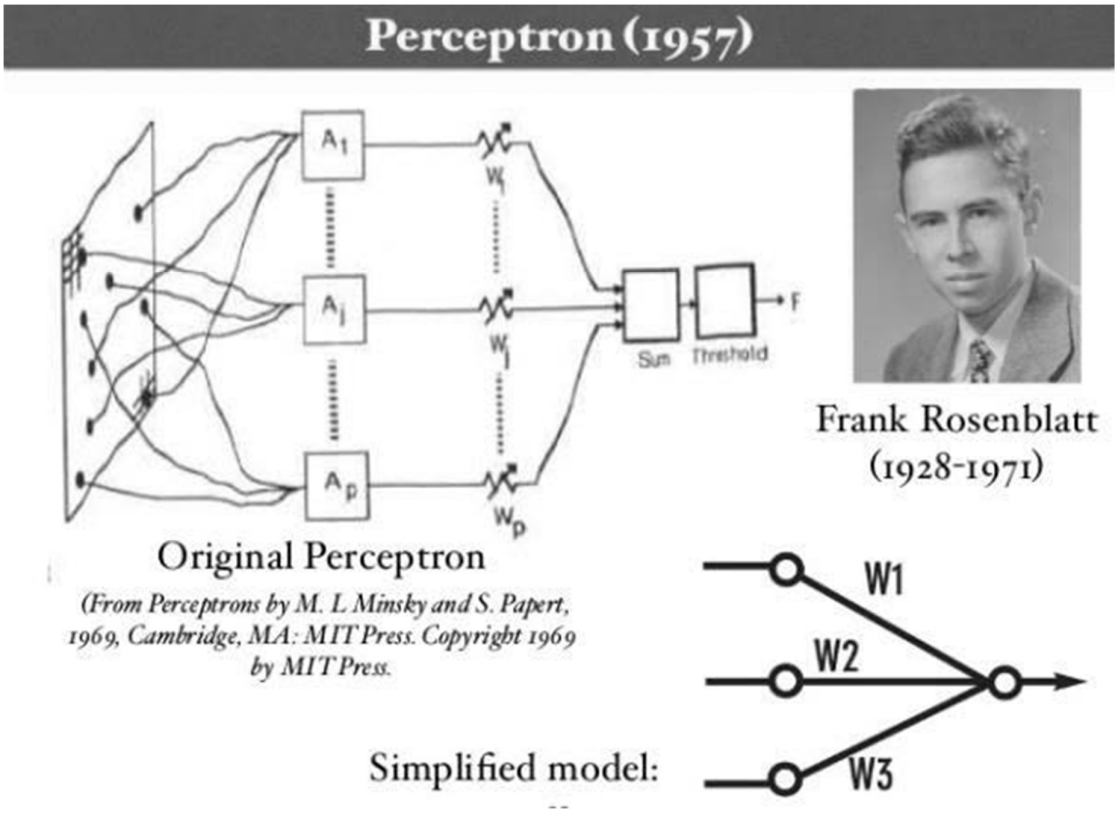
i. Single layer feed-forward network
ii. Multi-layer feed-forward network
iii. Single node with its own feedback
iv. Single layer recurrent network
v. Multi-layer recurrent network

6.1.2 Perceptron

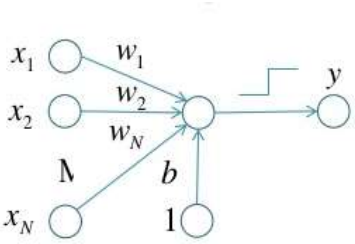
- The perceptron is the simplest form of neural network is able to classify data into two classes (Input and Output).
- It consists of single neuron with a number of adjustable weights.
- A single artificial neuron that compute its weighted input and uses a threshold activation function.
- It is also called a TLU (Threshold Logic Unit).



- Perceptron was introduced by Frank Rosenblatt in 1957. He proposed a Perceptron learning rule based on the original MCP neuron.
- A Perceptron is an algorithm for supervised learning of binary classifiers. This algorithm enables neurons to learn and processes elements in the training set one at a time.
- There are two types of Perceptrons: Single layer and Multilayer.
- **Single layer Perceptrons** can learn only linearly separable patterns.
- **Multilayer Perceptrons** or **feedforward neural networks** with two or more layers have the greater processing power.
- The Perceptron algorithm learns the weights for the input signals in order to draw a linear decision boundary.
- This enables you to distinguish between the two linearly separable classes +1 and -1.



Perceptron Original Research Paper By : Frank Rosenblatt



decision function

$\hat{f}(\bar{x}) = \text{sign}(\bar{w} \cdot \bar{x} + b)$

$\bar{x} = (x_1, x_2, \text{K}, x_N)$ features of the customer $\bar{x}$

approve credit if $\sum_{i=1}^{N} w_i x_i >$ threshold

deny credit if $\sum_{i=1}^{N} w_i x_i <$ threshold

Frank Rosenblatt

input: $\text{T} = \{(\bar{x}_i, y_i) \mid i = \overline{1, p}\} \subset \mathbf{R}^N \times \{+1, -1\}$

$\bar{w} \leftarrow \bar{0}, b \leftarrow 0$

repeat

  for i=1 **to** p

    if $\text{sign}(\bar{w} \cdot \bar{x}_i + b) \neq y_i$ **then**

      $\bar{w} \leftarrow \bar{w} + y_i \bar{x}_i$

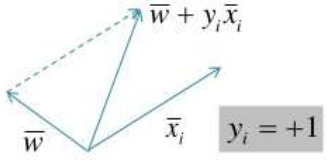
      $b \leftarrow b + y_i$

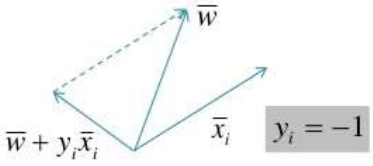
    end if

  end for

until $\text{sign}(\bar{w} \cdot \bar{x}_j + b) = y_j, \forall j = \overline{1, p}$

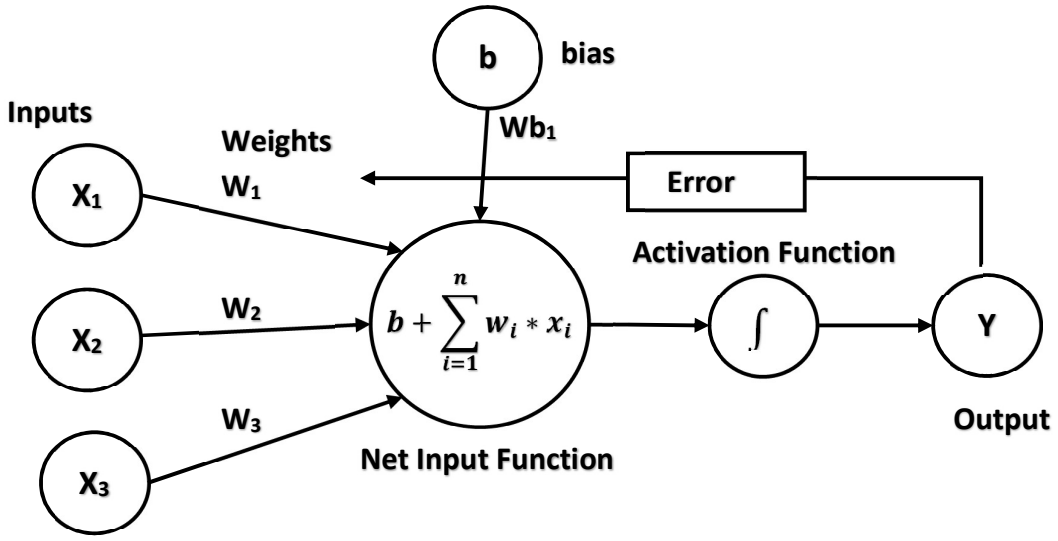
return $\bar{w}, b$






CORNELL AERONAUTICAL LABORATORY, INC.

Perceptron Learning Rule:



Perceptron Learning Rule states that the algorithm would automatically learn the optimal weight coefficients. The input features are then multiplied with these weights to determine if a neuron fire or not.

The Perceptron receives multiple input signals, and if the sum of the input signals exceeds a certain threshold, it either outputs a signal or does not return an output. In the context of supervised learning and classification, this can then be used to predict the class of a sample.

Perceptron Function:

Perceptron is a function that maps its input "x," which is multiplied with the learned weight coefficient; an output value "f(x)"is generated.

$$f(x) = \begin{cases} 1, & if\ w.x + b > 0 \\ 0, & if\ w.x + b \leq 0 \end{cases}$$

In the equation given above:

"w" = vector of real-valued weights

"b" = bias (an element that adjusts the boundary away from origin without any dependence on the input value)

"x" = vector of input x values

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

"n" = number of inputs to the Perceptron

The output can be represented as "1" or "0." It can also be represented as "1" or "-1" depending on which activation function is used.

What is Logic Gate?

Logic gates are the building blocks of a digital system, especially neural network. In short, they are the electronic circuits that help in addition, choice, negation, and combination to form complex circuits.

Using the logic gates, Neural Networks can learn on their own without you having to manually code the logic. Most logic gates have two inputs and one output.

Each terminal has one of the two binary conditions, low (0) or high (1), represented by different voltage levels. The logic state of a terminal changes based on how the circuit processes data.

Based on this logic, logic gates can be categorized into seven types:

1. AND
2. NAND
3. OR
4. NOR
5. NOT
6. XOR
7. XNOR

First, we need to know that the Perceptron algorithm states that:

Prediction (y`) = 1 if Wx+b > 0 and 0 if Wx+b ≤ 0

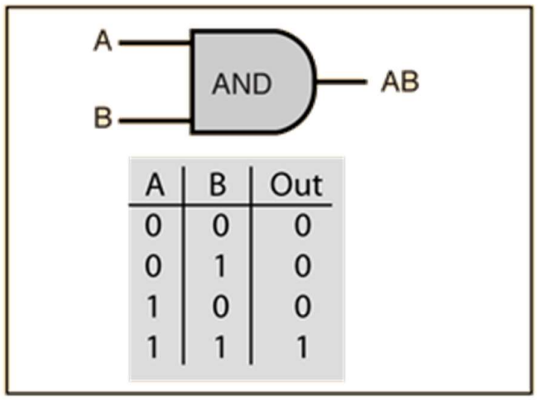
Also, the steps in this method are very similar to how Neural Networks learn, which is as follows;

- Initialize weight values and bias
- Forward Propagate
- Check the error
- Backpropagate and Adjust weights and bias
- Repeat for all training examples

Implement Basic Logic Gates with Perceptron:

AND Gate:

From our knowledge of logic gates, we know that an AND logic table is given by the diagram below:



A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

The question is, what are the weights and bias for the AND perceptron?

First, we need to understand that the output of an AND gate is 1 only if both inputs (in this case, x1 and x2) are 1.

Row 1

- From $W_1*X_1+W_2*X_2+b$, initializing W_1, W_2, as 1 and b as –1, we get;

$$= \boldsymbol{X_1(1) + X_2(1) + (-1)}$$

- Passing the first row of the AND logic table (**X_1=0, X_2=0**), we get;

$$= \mathbf{0(1) + 0(1) + (-1)}$$

$$= \mathbf{0 + 0 - 1 = -1}$$

- From the Perceptron rule, if $W * X + b \leq 0$, then y`= 0. Therefore, this row is correct, and no need for Backpropagation.

Row 2

- From $W_1*X_1+W_2*X_2+b$, initializing W_1, W_2, as 1 and b as –1, we get;

$$= \boldsymbol{X_1(1) + X_2(1) + (-1)}$$

- Passing the first row of the AND logic table (**X_1=0, X_2=1**), we get;

$$= \mathbf{0}(\mathbf{1}) + \mathbf{1}(\mathbf{1}) + (-\mathbf{1})$$

$$= \mathbf{0} + \mathbf{1} - \mathbf{1} = \mathbf{0}$$

- From the Perceptron rule, if $W * X + b \leq 0$, then y`= 0. Therefore, this row is correct, and no need for Backpropagation.

Row 3

- From W_1*X_1+W_2*X_2+b, initializing W_1, W_2, as 1 and b as –1, we get;

$$= \boldsymbol{X_1}(\mathbf{1}) + \boldsymbol{X_2}(\mathbf{1}) + (-\mathbf{1})$$

- Passing the first row of the AND logic table (**X_1=1, X_2=0**), we get;

$$= \mathbf{1}(\mathbf{1}) + \mathbf{0}(\mathbf{1}) + (-\mathbf{1})$$

$$= \mathbf{1} + \mathbf{0} - \mathbf{1} = \mathbf{0}$$

- From the Perceptron rule, if $W * X + b \leq 0$, then y`= 0. Therefore, this row is correct, and no need for Backpropagation.

Row 4

- From W_1*X_1+W_2*X_2+b, initializing W_1, W_2, as 1 and b as –1, we get;

$$= \boldsymbol{X_1}(\mathbf{1}) + \boldsymbol{X_2}(\mathbf{1}) + (-\mathbf{1})$$

- Passing the first row of the AND logic table (**X_1=1, X_2=1**), we get;

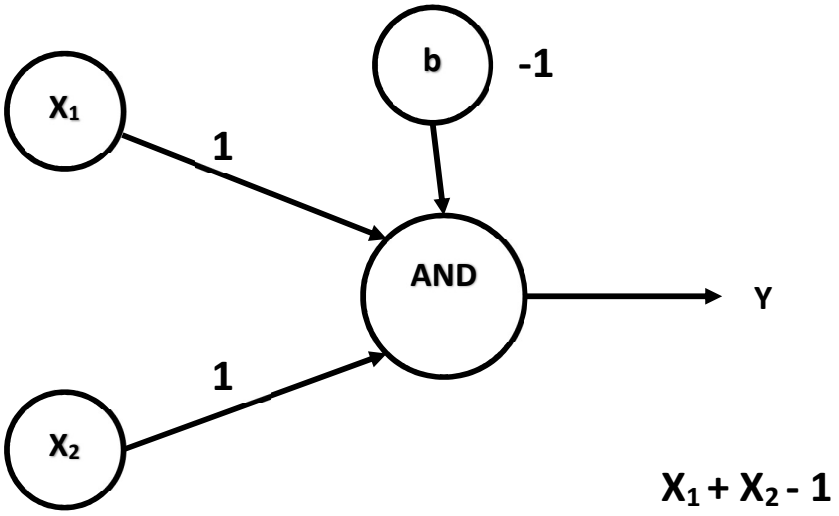
$$= \mathbf{1}(\mathbf{1}) + \mathbf{1}(\mathbf{1}) + (-\mathbf{1})$$

$$= \mathbf{1} + \mathbf{1} - \mathbf{1} = \mathbf{1}$$

- From the Perceptron rule, if $W * X + b > 0$, then y`= 1. Therefore, this row is correct, and no need for Backpropagation.

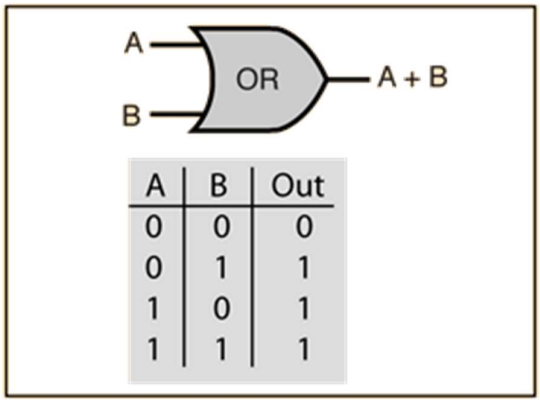
Therefore, we can conclude that the model to achieve an AND gate, using the Perceptron algorithm is;

$$= \boldsymbol{X_1} + \boldsymbol{X_2} - \mathbf{1}$$



OR Gate:

From our knowledge of logic gates, we know that an OR logic table is given by the diagram below:



A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1

From the diagram, the OR gate is 0 only if both inputs are 0.

Row 1

- From $W_1*X_1+W_2*X_2+b$, initializing W_1, W_2, as 1 and b as –1, we get;

$$= \boldsymbol{X_1(1) + X_2(1) + (-1)}$$

- Passing the first row of the OR logic table ($\mathbf{X_1=0, X_2=0}$), we get;

$$= \mathbf{0(1) + 0(1) + (-1)}$$

$$= \mathbf{0} + \mathbf{0} - \mathbf{1} = \mathbf{-1}$$

- From the Perceptron rule, if $W * X + b \leq 0$, then y`= 0. Therefore, this row is correct, and no need for Backpropagation.

Row 2

- From $W_1*X_1+W_2*X_2+b$, initializing W_1, W_2, as 1 and b as –1, we get;

$$= \boldsymbol{X_1}(\mathbf{1}) + \boldsymbol{X_2}(\mathbf{1}) + (\mathbf{-1})$$

- Passing the first row of the OR logic table (**X_1=0, X_2=1**), we get;

$$= \mathbf{0}(\mathbf{1}) + \mathbf{1}(\mathbf{1}) + (\mathbf{-1})$$

$$= \mathbf{0} + \mathbf{1} - \mathbf{1} = \mathbf{0}$$

- From the Perceptron rule, if $W * X + b \leq 0$, then y`= 0. Therefore, this row is incorrect, and need for Backpropagation.
- So, we want values that will make inputsX_1=0 and X_2=1 give y` a value of 1. If we change W_2 to 2, we have;

$$= \boldsymbol{X_1}(\mathbf{1}) + \boldsymbol{X_2}(\mathbf{2}) + (\mathbf{-1})$$

$$= \mathbf{0}(\mathbf{1}) + \mathbf{1}(\mathbf{2}) + (\mathbf{-1})$$

$$= \mathbf{0} + \mathbf{2} - \mathbf{1} = \mathbf{1}$$

- From the Perceptron rule, this is correct for both row1 and row2 no any change in row1.

Row 3

- From $W_1*X_1+W_2*X_2+b$, initializing W_1, W_2, as 1 and b as –1, we get;

$$= \boldsymbol{X_1}(\mathbf{1}) + \boldsymbol{X_2}(\mathbf{1}) + (\mathbf{-1})$$

- Passing the first row of the OR logic table (**X_1=1, X_2=0**), we get;

$$= \mathbf{1}(\mathbf{1}) + \mathbf{0}(\mathbf{1}) + (\mathbf{-1})$$

$$= \mathbf{1} + \mathbf{0} - \mathbf{1} = \mathbf{0}$$

- From the Perceptron rule, if $W * X + b \leq 0$, then y`= 0. Therefore, this row is incorrect, and need for Backpropagation.
- Since it is similar to that of row 2. We can just change W_1 to 2, we have;

$$= \boldsymbol{X_1}(\mathbf{2}) + \boldsymbol{X_2}(\mathbf{1}) + (\mathbf{-1})$$

$$= \mathbf{1}(\mathbf{2}) + \mathbf{0}(\mathbf{1}) + (-\mathbf{1})$$

$$= \mathbf{2} + \mathbf{0} - \mathbf{1} = \mathbf{1}$$

- From the Perceptron rule, this is correct for both row1, row2 and row3.

Row 4

- From $W_1*X_1+W_2*X_2+b$, initializing W_1, W_2, as 1 and b as –1, we get;

$$= \boldsymbol{X_1}(\mathbf{1}) + \boldsymbol{X_2}(\mathbf{1}) + (-\mathbf{1})$$

- Passing the first row of the OR logic table (**X_1=1, X_2=1**), we get;

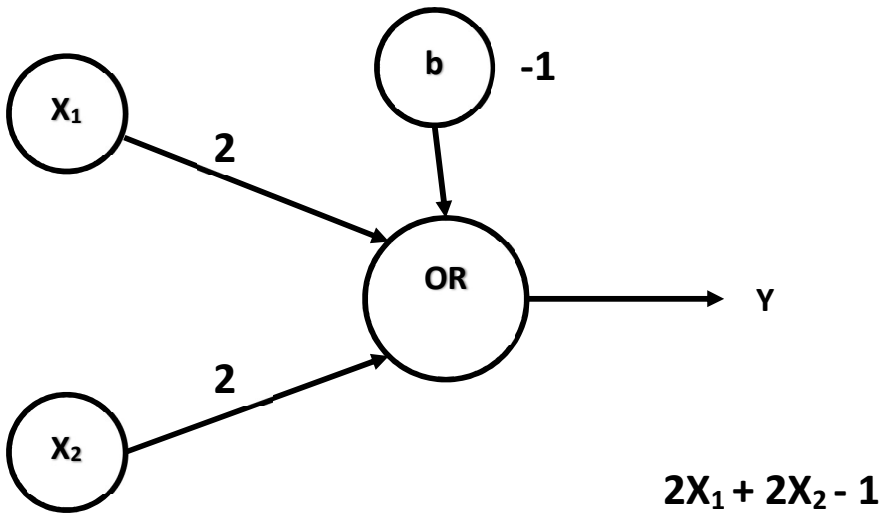
$$= \mathbf{1}(\mathbf{1}) + \mathbf{1}(\mathbf{1}) + (-\mathbf{1})$$

$$= \mathbf{1} + \mathbf{1} - \mathbf{1} = \mathbf{1}$$

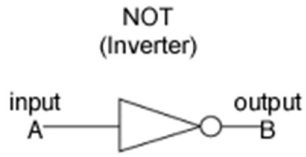
- From the Perceptron rule, if $W * X + b > 0$, then y`= 1. Therefore, this row is correct, and no need for Backpropagation.

Therefore, we can conclude that the model to achieve an OR gate, using the Perceptron algorithm is;

$$= \mathbf{2}\boldsymbol{X_1} + \mathbf{2}\boldsymbol{X_2} - \mathbf{1}$$



NOT Gate:



A	B
0	1
1	0

From the diagram, the output of a NOT gate is the inverse of a single input.

Row 1

- From W_1*X_1+b, initializing W_1 as 1 (since single input), and b as –1, we get;

$$= \boldsymbol{X_1(1) + (-1)}$$

- Passing the first row of the NOT logic table (x1=0), we get;

$$= \mathbf{0(1) + (-1)}$$

$$= \mathbf{0 - 1 = -1}$$

- From the Perceptron rule, if $W * X + b \leq 0$, then y`=0. This row is incorrect, as the output is 1 for the NOT gate.
- So, we want values that will make input x1=0 to give y` a value of 1. If we change b to +1, we have;

$$= \mathbf{0(1) + (+1)}$$

$$= \mathbf{0 + 1 = 1}$$

- From the Perceptron rule, this works.

Row 2

- From W_1*X_1+b, initializing W_1 as 1 (since single input), and b as +1, we get;

$$= \boldsymbol{X_1(1) + (+1)}$$

- Passing the second row of the NOT logic table (x1=1), we get;

$$= \mathbf{1(1) + (+1)}$$

$$= \mathbf{1 + 1 = 2}$$

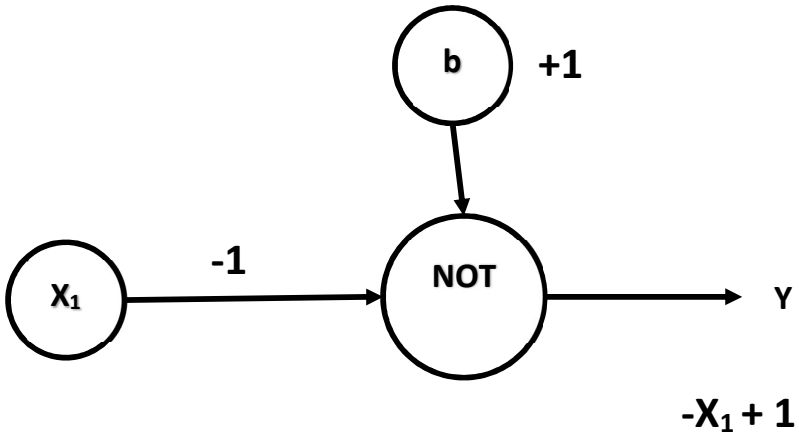
- From the Perceptron rule, if $W * X + b > 0$, then y`=1. This row is so incorrect, as the output is 0 for the NOT gate.
- So, we want values that will make input x1=1 to give y` a value of 0. If we change W_1 to –1, we have;

$$= -\mathbf{1}(\mathbf{1}) + (+\mathbf{1})$$
$$= -\mathbf{1} + \mathbf{1} = \mathbf{0}$$

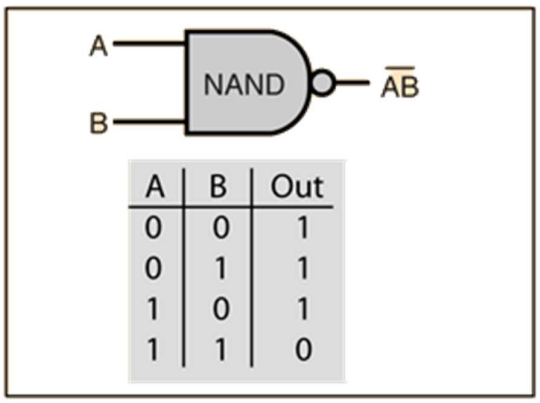
- From the Perceptron rule, if $W * X + b \leq 0$, then y`=0. Therefore, this works (for both row 1 and row 2).

Therefore, we can conclude that the model to achieve a NOT gate, using the Perceptron algorithm is;

$$= -\boldsymbol{X_1} + \mathbf{1}$$



NAND Gate:



From the diagram, the NAND gate is 0 only if both inputs are 1.

Row 1

- From $W_1*X_1+W_2*X_2+b$, initializing W_1, W_2, as 1 and b as –1, we get;

$$= \boldsymbol{X_1(1) + X_2(1) + (-1)}$$

- Passing the first row of the NAND logic table (**X_1=0, X_2=0**), we get;

$$= \mathbf{0(1) + 0(1) + (-1)}$$

$$= \mathbf{0 + 0 - 1 = -1}$$

- From the Perceptron rule, if $W * X + b \leq 0$, then y`= 0. Therefore, this row is incorrect, as the output is 1 for the NAND gate and need for Backpropagation.

- So, we want values that will make input x1=0 and x2 = 0 to give y` a value of 1. If we change b to 1, we have;

$$= \mathbf{0(1) + 0(1) + (+1)}$$

$$= \mathbf{0 + 0 + 1 = 1}$$

- From the Perceptron rule, if $W * X + b > 0$, then y`= 1. Therefore, this row is correct

Row 2

- From $W_1*X_1+W_2*X_2+b$, initializing W_1, W_2, as 1 and b as +1, we get;

$$= \boldsymbol{X_1(1) + X_2(1) + (+1)}$$

- Passing the first row of the NAND logic table (**X_1=0, X_2=1**), we get;

$$= \mathbf{0(1) + 1(1) + (+1)}$$

$$= \mathbf{0 + 1 + 1 = 2}$$

- From the Perceptron rule, if $W * X + b > 0$, then y`= 1. Therefore, this row is correct, and no need for Backpropagation.

Row 3

- From $W_1*X_1+W_2*X_2+b$, initializing W_1, W_2, as 1 and b as +1, we get;

$$= \boldsymbol{X_1(1) + X_2(1) + (+1)}$$

- Passing the first row of the NAND logic table (**X_1=1, X_2=0**), we get;

$$= \mathbf{1(1) + 0(1) + (+1)}$$

$$= \mathbf{1 + 0 + 1 = 2}$$

- From the Perceptron rule, if $W * X + b > 0$, then y`= 1. Therefore, this row is correct, and no need for Backpropagation.

Row 4

- From $W_1*X_1+W_2*X_2+b$, initializing W_1, W_2, as 1 and b as +1, we get;

$$= \boldsymbol{X_1(1) + X_2(1) + (+1)}$$

- Passing the first row of the NAND logic table ($\mathbf{X_1=1, X_2=1}$), we get;

$$= \mathbf{1(1) + 1(1) + (+1)}$$

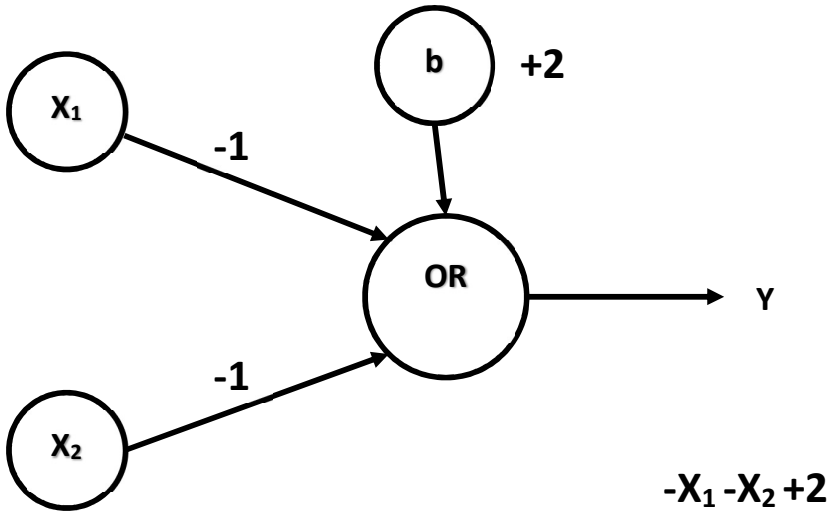
$$= \mathbf{1 + 1 + 1 = 3}$$

- From the Perceptron rule, if $W * X + b > 0$, then y`= 1. It is incorrect.
- This is not the expected output, as the output is 0 for a NAND combination of X_1=1 and X_2=1.
- Changing values of W_1 and W_2 to -1, and value of b to 2, we get;

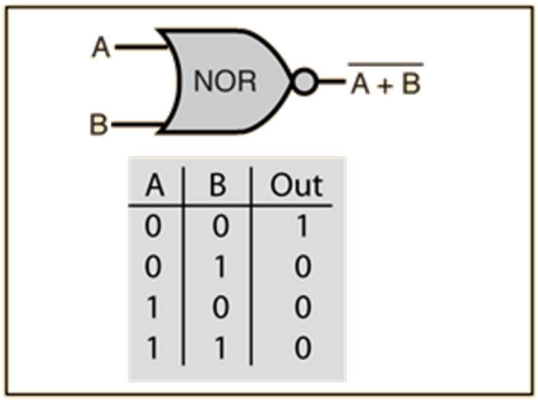
$$= \mathbf{1(-1) + 1(-1) + (+2)}$$
$$= \mathbf{-1 - 1 + 2 = 0}$$

- From the Perceptron rule, if $W * X + b \leq 0$, then y`= 0. Therefore, this row is correct.
- It works for all rows.
- Therefore, we can conclude that the model to achieve a NAND gate, using the Perceptron algorithm is;

$$= \boldsymbol{-X_1 - X_2 + 2}$$



NOR Gate:



A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

From the diagram, the NOR gate is 1 only if both inputs are 0.

Row 1

- From $W_1*X_1+W_2*X_2+b$, initializing W_1, W_2, as 1 and b as –1, we get;
$$= \boldsymbol{X_1(1) + X_2(1) + (-1)}$$

- Passing the first row of the NOR logic table (**X_1=0, X_2=0**), we get;
$$= \mathbf{0(1) + 0(1) + (-1)}$$
$$= \mathbf{0 + 0 - 1 = -1}$$

- From the Perceptron rule, if $W * X + b \leq 0$, then y`= 0. Therefore, this row is incorrect, as the output is 1 for the NOR gate and need for Backpropagation.

- So, we want values that will make input x1=0 and x2 = 0 to give y` a value of 1. If we change b to 1, we have;
$$= \mathbf{0(1) + 0(1) + (+1)}$$
$$= \mathbf{0 + 0 + 1 = 1}$$

- From the Perceptron rule, if $W * X + b > 0$, then y`= 1. Therefore, this row is correct.

Row 2

- From $W_1*X_1+W_2*X_2+b$, initializing W_1, W_2, as 1 and b as +1, we get;
$$= \boldsymbol{X_1(1) + X_2(1) + (+1)}$$

- Passing the first row of the NOR logic table (**X_1=0, X_2=1**), we get;
$$= \mathbf{0(1) + 1(1) + (+1)}$$
$$= \mathbf{0 + 1 + 1 = 2}$$

- From the Perceptron rule, if $W * X + b > 0$, then y`= 1. Therefore, this row is incorrect, and need for Backpropagation.

- So, we want values that will make input X_1=0 and X_2 = 1 to give y` a value of 0. If we change W_2 to –1, we have;

$$\mathbf{0(1) + 1(-1) + (+1)}$$
$$\mathbf{= 0 - 1 + 1 = 0}$$

- From the Perceptron rule, this is valid for both row 1 and row 2.

Row 3

- From W_1*X_1+W_2*X_2+b, initializing W_1, W_2, as 1 and b as +1, we get;

$$\mathbf{= X_1(1) + X_2(1) + (+1)}$$

- Passing the first row of the NOR logic table (**X_1=1, X_2=0**), we get;

$$\mathbf{= 1(1) + 0(1) + (+1)}$$

$$\mathbf{= 1 + 0 + 1 = 2}$$

- From the Perceptron rule, if $W * X + b > 0$, then y`= 1. Therefore, this row is incorrect, and need for Backpropagation.

- So, we want values that will make input X_1=0 and X_2 = 1 to give y` a value of 0. If we change W_1 to –1, we have;

$$\mathbf{1(-1) + 0(1) + (+1)}$$
$$\mathbf{= 1 - 0 + 1 = 0}$$

- From the Perceptron rule, this is valid for both row 1, row 2 and row 3.

Row 4

- From W_1*X_1+W_2*X_2+b, initializing W_1, W_2, as 1 and b as +1, we get;

$$\mathbf{= X_1(1) + X_2(1) + (+1)}$$

- Passing the first row of the NOR logic table (**X_1=1, X_2=1**), we get;

$$\mathbf{= 1(1) + 1(1) + (+1)}$$

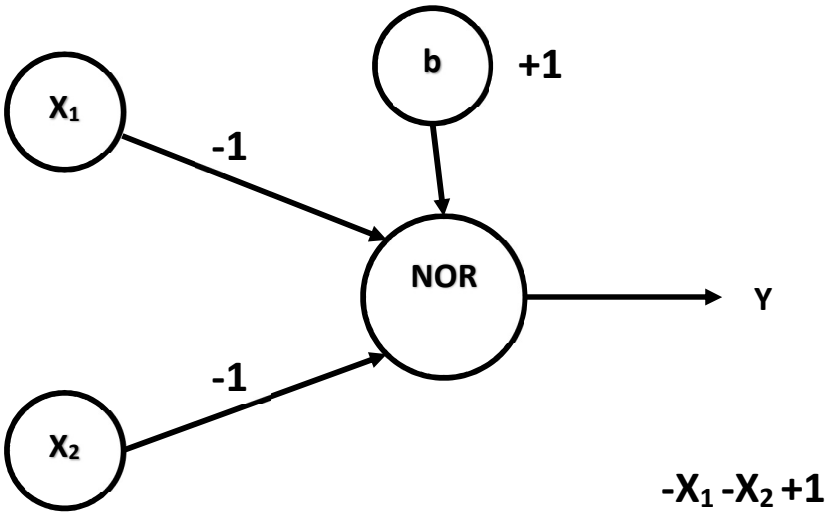
$$\mathbf{= 1 + 1 + 1 = 3}$$

- From the Perceptron rule, if $W * X + b > 0$, then y`= 1. It is incorrect.
- This is not the expected output, as the output is 0 for a NOR combination of X_1=1 and X_2=1.
- Changing values of W_1 and W_2 to -1, we get;

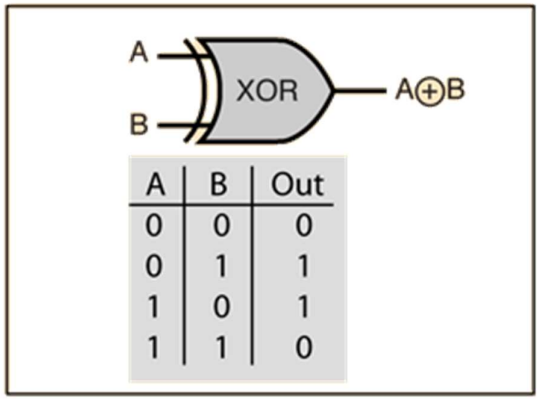
$$= \mathbf{1}(-\mathbf{1}) + \mathbf{1}(-\mathbf{1}) + (+\mathbf{1})$$
$$= -\mathbf{1} - \mathbf{1} + \mathbf{1} = -\mathbf{1}$$

- From the Perceptron rule, if $W * X + b \leq 0$, then y`= 0. Therefore, this row is correct.
- It works for all rows.
- Therefore, we can conclude that the model to achieve a NOR gate, using the Perceptron algorithm is;

$$= -\boldsymbol{X_1} - \boldsymbol{X_2} + \mathbf{1}$$



XOR Gate:



A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

If a specific type of gate is not available, a circuit that implements the same function can be constructed from other available gates.

A circuit implementing an XOR function can be trivially constructed from an XNOR gate followed by a NOT gate.

If we consider the expression, $(\boldsymbol{A}.\overline{\boldsymbol{B}}) + (\overline{\boldsymbol{A}}.\boldsymbol{B})$ we can construct an XOR gate circuit directly using AND, OR and NOT gates. However, this approach requires five gates of three different kinds.

As alternative, if different gates are available we can apply Boolean algebra to transform $(\boldsymbol{A}.\overline{\boldsymbol{B}}) + (\overline{\boldsymbol{A}}.\boldsymbol{B}) \equiv (\boldsymbol{A}+\boldsymbol{B}).(\overline{\boldsymbol{A}}+\overline{\boldsymbol{B}})$ as stated above, and apply de Morgan's Law to the last term to get $(\boldsymbol{A}+\boldsymbol{B}).(\overline{\boldsymbol{A}.\ \boldsymbol{B}})$ which can be implemented using only three gates

How this expression $(\boldsymbol{A}+\boldsymbol{B}).(\overline{\boldsymbol{A}}+\overline{\boldsymbol{B}})$ is transformed?

The Boolean representation of an XOR gate is;

$$= (\boldsymbol{A}.\overline{\boldsymbol{B}}) + (\overline{\boldsymbol{A}}.\boldsymbol{B})$$

We first simplify the Boolean expression

$= \boldsymbol{X_1}\overline{\boldsymbol{X_2}} + \overline{\boldsymbol{X_1}}\boldsymbol{X_2}$ **(Apply De-Morgan's Law)**

$= \overline{\boldsymbol{A}.\overline{\boldsymbol{B}}}.\ \overline{\overline{\boldsymbol{A}}.\boldsymbol{B}}$ **(Apply De-Morgan's Law)**

$= \overline{\boldsymbol{A}} + \overline{\overline{\boldsymbol{B}}}.\ \overline{\overline{\boldsymbol{A}}} + \overline{\boldsymbol{B}}$

$= (\overline{\boldsymbol{A}}+\boldsymbol{B}).\ (\boldsymbol{A}+\overline{\boldsymbol{B}})$ **(Apply Distributive Law)**

$= \overline{\boldsymbol{A}}(\boldsymbol{A}+\overline{\boldsymbol{B}}) + \boldsymbol{B}(\boldsymbol{A}+\overline{\boldsymbol{B}})$ **(Apply Distributive Law)**

$= \overline{\boldsymbol{A}}.\boldsymbol{A} + \overline{\boldsymbol{A}}.\overline{\boldsymbol{B}} + \boldsymbol{B}.\boldsymbol{A} + \boldsymbol{B}.\overline{\boldsymbol{B}}$ **(Apply Complement Law)** $\overline{\boldsymbol{A}}.\boldsymbol{A} = \boldsymbol{0}$

$= \boldsymbol{0} + \overline{\boldsymbol{A}}.\overline{\boldsymbol{B}} + \boldsymbol{B}.\boldsymbol{A} + \boldsymbol{0}$

$= \overline{\boldsymbol{A}}.\overline{\boldsymbol{B}} + \boldsymbol{B}.\boldsymbol{A}$ **(Apply De-Morgan's Law)**

$= \overline{(\overline{\boldsymbol{A}}.\overline{\boldsymbol{B}}).(\boldsymbol{B}.\boldsymbol{A})}$

$= (\overline{\overline{\boldsymbol{A}}} + \overline{\overline{\boldsymbol{B}}}).(\overline{\boldsymbol{B}.\boldsymbol{A}})$ **(Apply Commutative Law) B.A = A.B**

$= (\boldsymbol{A}+\boldsymbol{B}).(\overline{\boldsymbol{A}.\boldsymbol{B}})$

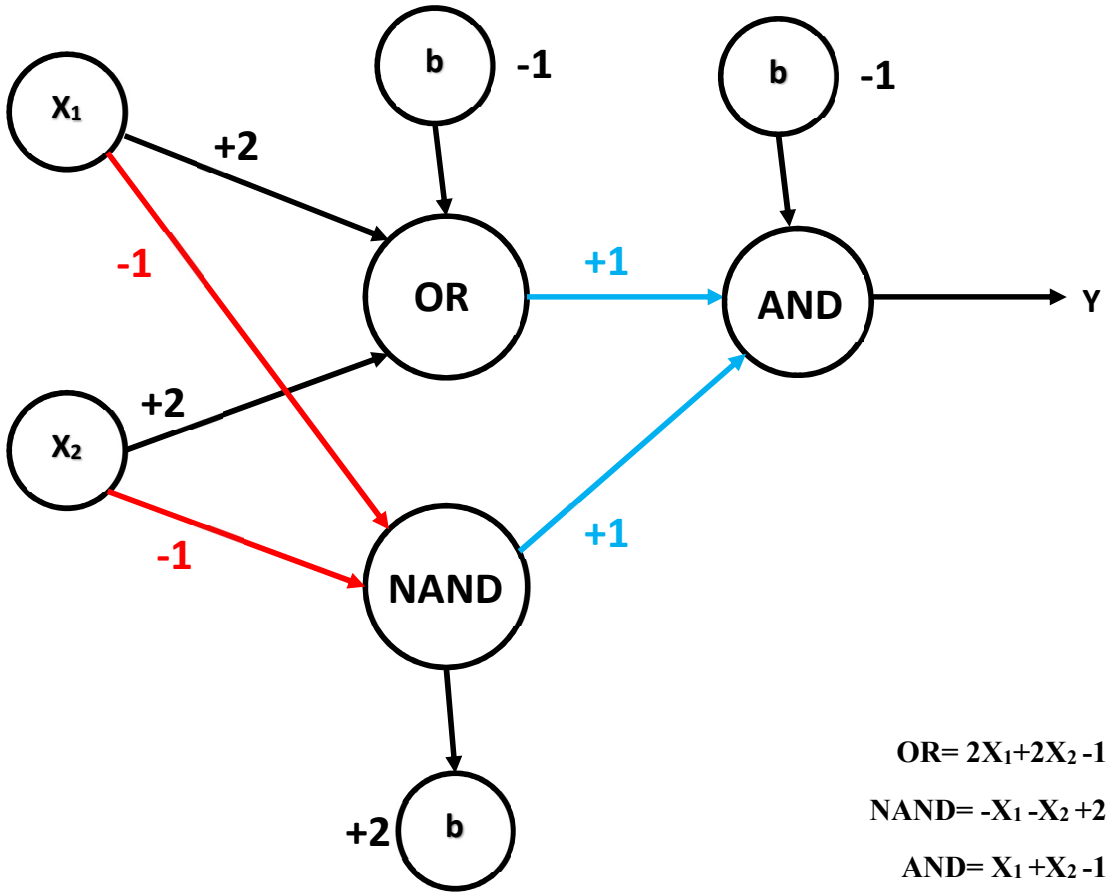
$= X_1\overline{X_2} + \overline{X_1}X_2$ **So, this can be equivalent as,**

$= (X_1 + X_2)\ .\ (\overline{X_1\ .\ X_2})$

From the simplified expression, we can say that the XOR gate consists of an OR gate ($X_1 + X_2$), a NAND gate ($\overline{X_1\ .\ X_2}$) and an AND gate ($(X_1 + X_2)\ .\ (\overline{X_1\ .\ X_2})$)

This means we will have to combine 3 perceptrons:

- OR ($2X_1 + 2X_2 - 1$)
- NAND ($-X_1 - X_2 + 1$)
- AND ($X_1 + X_2 - 1$)



OR= $2X_1+2X_2$ -1

NAND= $-X_1 -X_2$ +2

AND= $X_1 +X_2$ -1

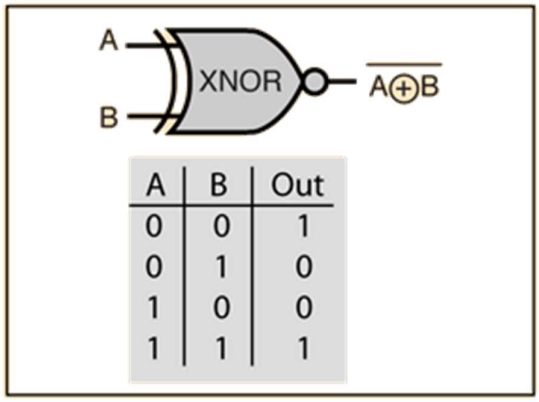
Row 1 (X_1= 0, X_2= 0)		**OR(X_1)= 0, NAND(X_2)= 1**
OR ($2X_1+2X_2-1$)	**NAND ($-X_1-X_2+2$)**	**AND (X_1+X_2-1)**
2*0+2*0-1= -1	**-0-0+2= +2**	**0+1-1= 0**
From the perceptron rule, if $W * X + b \leq 0$, then **y`= 0**	From the perceptron rule, if $W * X + b > 0$, then **y`= 1**	From the perceptron rule, if $W * X + b \leq 0$, then **y`= 0**
This row is correct as the output is 0 for the XOR gate (0 & 0 is 0)		

Row 2 (X_1= 0, X_2= 1)		**OR(X_1)= 1, NAND(X_2)= 1**
OR ($2X_1+2X_2-1$)	**NAND ($-X_1-X_2+2$)**	**AND (X_1+X_2-1)**
2*0+2*1-1= +1	**-0-1+2= +1**	**1+1-1= 1**
From the perceptron rule, if $W * X + b > 0$, then **y`= 1**	From the perceptron rule, if $W * X + b > 0$, then **y`= 1**	From the perceptron rule, if $W * X + b > 0$, then **y`= 1**
This row is correct as the output is 0 for the XOR gate (0 & 1 is 1)		

Row 3 (X_1= 1, X_2= 0)		**OR(X_1)= 1, NAND(X_2)= 1**
OR ($2X_1+2X_2-1$)	**NAND ($-X_1-X_2+2$)**	**AND (X_1+X_2-1)**
2*1+2*0-1= +1	**-1-0+2= +1**	**1+1-1= 1**
From the perceptron rule, if $W * X + b > 0$, then **y`= 1**	From the perceptron rule, if $W * X + b > 0$, then **y`= 1**	From the perceptron rule, if $W * X + b > 0$, then **y`= 1**
This row is correct as the output is 0 for the XOR gate (1 & 0 is 1)		

Row 4 (X_1= 1, X_2= 1)		**OR(X_1)= 1, NAND(X_2)= 0**
OR ($2X_1+2X_2-1$)	**NAND ($-X_1-X_2+2$)**	**AND (X_1+X_2-1)**
2*1+2*1-1= +3	**-1-1+2= 0**	**1+0-1= 0**
From the perceptron rule, if $W * X + b > 0$, then **y`= 1**	From the perceptron rule, if $W * X + b \leq 0$, then **y`= 0**	From the perceptron rule, if $W * X + b \leq 0$, then **y`= 0**
This row is correct as the output is 0 for the XOR gate (1 & 1 is 0)		

XNOR Gate:



A	B	Out
0	0	1
0	1	0
1	0	0
1	1	1

Now that we are done with the necessary basic logic gates, we can combine them to give an XNOR gate. This XNOR gate is the complement or inverse of XOR gate.

Means, XOR= $(\boldsymbol{A} + \boldsymbol{B}) \,.\, (\overline{\boldsymbol{A} \,.\, \boldsymbol{B}})$ **i.e.** $\boldsymbol{XNOR} = \overline{\boldsymbol{XOR}}$

Then, XNOR= $\overline{(A + B) \,.\, (\overline{A \,.\, B})}$

$= (\overline{\boldsymbol{A} + \boldsymbol{B}}) + \left(\overline{\overline{\boldsymbol{A} \,.\, \boldsymbol{B}}}\right)$

The Boolean representation of an XNOR gate is;

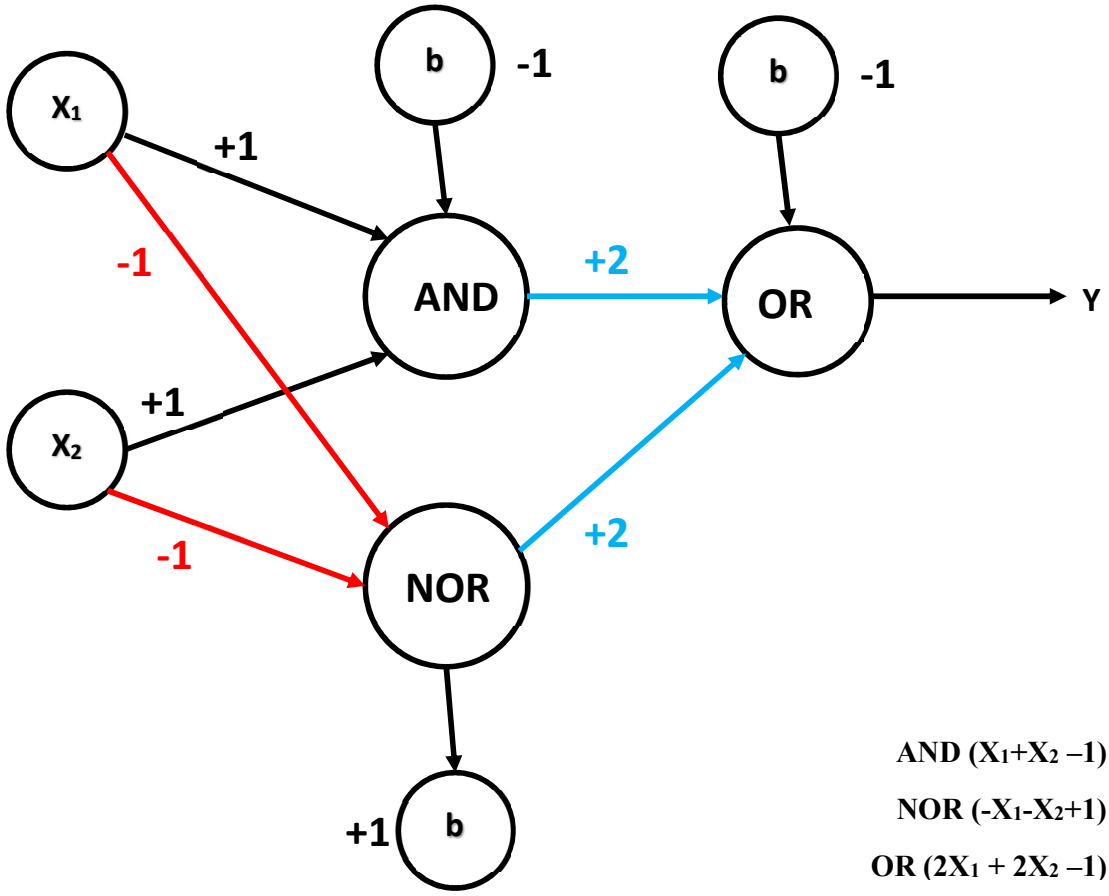
$= (\overline{\boldsymbol{A}} \,.\, \overline{\boldsymbol{B}}) + (\boldsymbol{A} \,.\, \boldsymbol{B})$

$= (\overline{\boldsymbol{X_1}} \,.\, \overline{\boldsymbol{X_2}}) + (\boldsymbol{X_1} \,.\, \boldsymbol{X_2})$

From the expression, we can say that the XNOR gate consists of an AND gate ($\boldsymbol{X_1} \,.\, \boldsymbol{X_2}$), a NOR gate ($\overline{\boldsymbol{X_1}} \,.\, \overline{\boldsymbol{X_2}}$), and an OR gate (($\overline{\boldsymbol{X_1}} \,.\, \overline{\boldsymbol{X_2}}$) + ($\boldsymbol{X_1} \,.\, \boldsymbol{X_2}$)).

This means we will have to combine 3 perceptrons:

- **AND (X_1+X_2-1)**
- **NOR ($-X_1-X_2+1$)**
- **OR ($2X_1 + 2X_2 - 1$)**



Row 1 (X_1= 0, X_2= 0)		**AND(X_1)= 0, NOR(X_2)= 1**
AND (X_1+X_2 –1)	**NOR (-X_1-X_2+1)**	**OR (2X_1 + 2X_2 –1)**
0+0-1= -1	**-0-0+1= +1**	**2*0+2*1-1= 1**
From the perceptron rule, if $W * X + b \leq 0$, then **y`= 0**	From the perceptron rule, if $W * X + b > 0$, then **y`= 1**	From the perceptron rule, if $W * X + b > 0$, then **y`= 1**
This row is correct as the output is 0 for the XNOR gate (0 & 0 is 1)		

Row 2 (X_1= 0, X_2= 1)		**AND(X_1)= 0, NOR(X_2)= 0**
AND (X_1+X_2 –1)	**NOR (-X_1-X_2+1)**	**OR (2X_1 + 2X_2 –1)**
0+1-1= 0	**-0-1+1= 0**	**2*0+2*0-1= -1**
From the perceptron rule, if $W * X + b \leq 0$, then **y`= 0**	From the perceptron rule, if $W * X + b \leq 0$, then **y`= 0**	From the perceptron rule, if $W * X + b \leq 0$, then **y`= 0**
This row is correct as the output is 0 for the XNOR gate (0 & 1 is 0)		

Row 3 (X_1= 1, X_2= 0)		**AND(X_1)= 0, NOR(X_2)= 0**
AND (X_1+X_2 –1)	**NOR (-X_1-X_2+1)**	**OR (2X_1 + 2X_2 –1)**
1+0-1= 0	**-1-0+1= 0**	**2*0+2*0-1= -1**
From the perceptron rule, if $W * X + b \leq 0$, then **y`= 0**	From the perceptron rule, if $W * X + b \leq 0$, then **y`= 0**	From the perceptron rule, if $W * X + b \leq 0$, then **y`= 0**
This row is correct as the output is 0 for the XNOR gate (1 & 0 is 0)		

Row 4 (X_1= 1, X_2= 1)		**AND(X_1)= 1, NOR(X_2)= 0**
AND (X_1+X_2 –1)	**NOR (-X_1-X_2+1)**	**OR (2X_1 + 2X_2 –1)**
1+1-1= +1	**-1-1+1= -1**	**2*1+2*0-1= +1**
From the perceptron rule, if $W * X + b > 0$, then **y`= 1**	From the perceptron rule, if $W * X + b \leq 0$, then **y`= 0**	From the perceptron rule, if $W * X + b > 0$, then **y`= 1**
This row is correct as the output is 0 for the XNOR gate (1 & 1 is 1)		

CONCLUSION

In conclusion, this is just a custom method of achieving this, there are many other ways and values you could use in order to achieve logic gates using perceptrons. For example;

AND (20X_1+20X_2–30)

OR (20X_1+20X_2–10)

NOT (-20X_1+10)

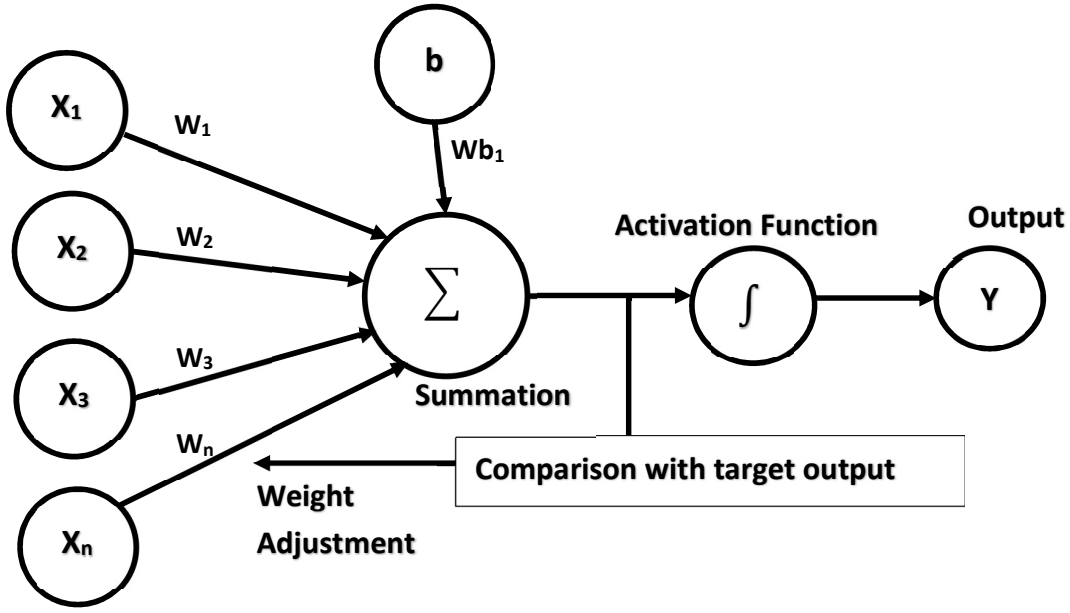
This will still work.

6.1.3 Adaline Network

- ADALINE stands for **ADA**ptive **LI**near **NE**uron or **ADA**ptive **LIN**ear **E**lement, is a network having a single linear unit.
- It was developed by Widrow and Hoff in 1960.
- Some important points about Adaline are as follows: -
    - It uses bipolar(+1 or -1) activation function.
    - It uses delta rule for training to minimize the Mean-Squared Error (MSE) between the actual output and the desired/ target output.
    - The weights and the bias are adjustable.

Architecture:

The basic structure of Adaline is similar to perceptron having an extra feedback loop with the help of which the actual output is compared with the desired/ target output. After comparison on the basis of training algorithm, the weights and bias will be updated.



Adaline Algorithm:

Step 1: Initialize the weights to small random values and select a learning rate, α

Step 2: For each input vector x, with target output t, set the inputs to S=(x-t)

Step 3: Compute the neuron inputs: $\boldsymbol{Y_{in} = b + \sum X_i W_i}$

Step 4: Use the delta rule to update the bias and weights:

$$\boldsymbol{b(new) = b(old) + \propto (t - Y_{in})}$$

$$\boldsymbol{W_i(new) = W_i(old) + \propto (t - Y_{in}) X_i}$$

Step 5: Stop if the largest weight change across all the training samples is less than a specified tolerance, otherwise cycle through the training set again.

The Learning Rate, $\propto$

- The performance of an ADALINE neuron depends heavily on the choice of the learning rate.
  - If it is too large the system will not converge
  - If it is too small the convergence will take too long
- Typically, $\propto$ is selected by trial and error
  - Typical range: $0.01 < \propto < 1.0$
  - Often start at 0.1
  - Sometimes it is suggested that:
    - $0.01 < n \propto < 1.0$
      Where n is the number of inputs

Running Adaline

- One unique feature of ADALINE is that its activation function is different for training and running
- When running ADALINE use the following:
  - Initialize the weights to those found during training
  - Compute the Net input
  - Apply the activation function

Neuron Input	Activation Function
$\boldsymbol{Y_{in} = b + \sum X_i W_i}$	$\boldsymbol{Y = \begin{cases} 1, & if\ Y_{in} \geq 0 \\ -1, & if\ Y_{in} < 0 \end{cases}}$

6.1.4 Multilayer Perceptron, Back Propagation

Multiple Layer Perceptron:

In simple a multilayer perceptron is a feedforward neural network with one or more hidden layers.

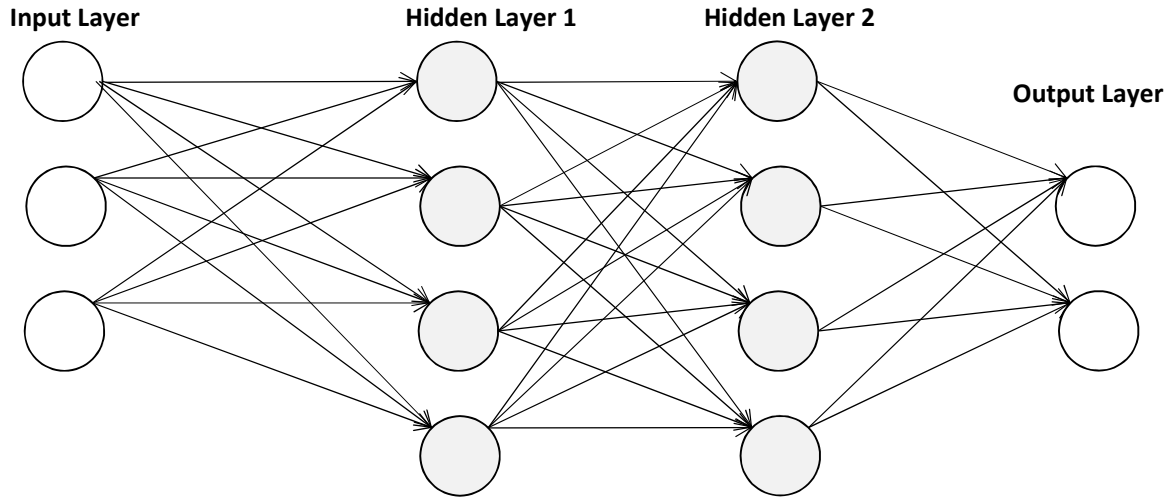
In the Multilayer perceptron, there can more than one linear layer (combinations of neurons). If we take the simple example the three-layer network, first layer will be the **input layer** and last will be **output layer** and middle layer will be called **hidden layer**. We feed our input data into the input layer and take the output from the output layer. We can increase the number of the hidden layer as much as we want, to make the model more complex according to our task.

So, in MLP network consists of an **input layer** of source neurons, at least one middle or **hidden layer** of computational neurons, and an **output layer** of computational neurons.

The input signals are propagated in a forward direction on a layer-by-layer basis.

A hidden layer "hides" its desired output. Neurons in the hidden layer cannot be observed through the input/output behavior of the network. There is no obvious way to know what the desired output of the hidden layer should be.

Perceptrons can be improved if it is placed in a multilayered network.



In logic gate XOR and XNOR are the example of multilayered perceptron network.

Back Propagation:

The term back propagation and its general use in neural networks was announced by Rumelhart, Hinton & Williams in 1986.

Backpropagation, short for "**backward propagation of errors,**" is an algorithm for supervised learning of artificial neural networks using gradient descent.

In machine learning, back propagation is widely used algorithm in training multi-layer perceptrons (artificial Neural Networks).

The back-propagation algorithm is used to update the neural network weights when they are not able to make the correct predictions. Hence, we should train the neural network before applying backpropagation.

Learning in a multilayer network proceeds the same way as for a perceptron.

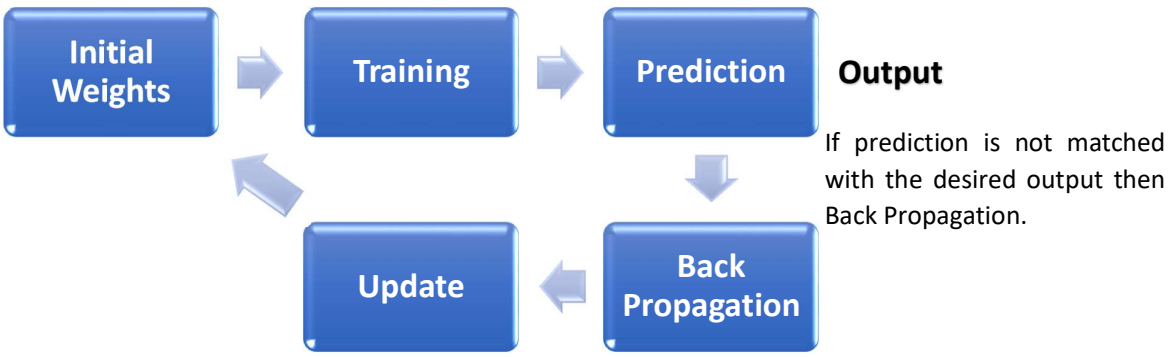
A training set of input patterns is presented to the network.

The network computes its output pattern, and if there is error or in other words a difference between actual output and desired output patterns the weights are adjusted to reduce the error.

In a back-propagation neural network, the training algorithm has two phases.

First, a training input pattern is presented to the network input layer. The network propagates the input pattern from layer to layer until the output pattern is generated by the output layer.

If the pattern is different from the desired output, an error is calculated and then propagated backward through the network from output layer to the input layer. The weights are modified as the error is propagated.



Back Propagation Training Algorithm:

Step 1: Initialization of weight and bias

Step 2: Feed Forward

- Caluclate sum of Products

$$s = b + \sum_{i=1}^{n} X_i * W_i$$

- Activation Function

We can use Sigmoid Activation Function

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

Step 3: Back Propagation of Error

- Prediction Error

$$E = \frac{1}{2}(desired\ output - predicated\ output)^2$$

Step 4: Updation of Weight and Bias

We can use weight update equation:

$$w_{new} = w_{old} + \propto [d - y] . X$$

w_{new} = New Updated Weights.

w_{old} = Current Weights.

$\propto$ = Network Learning Rate or η (Eta)

Assume any learning rate between 0 to 1 any positive value, default vlue is 0.1 or 0.01

d = Desired output

y = Predicted output

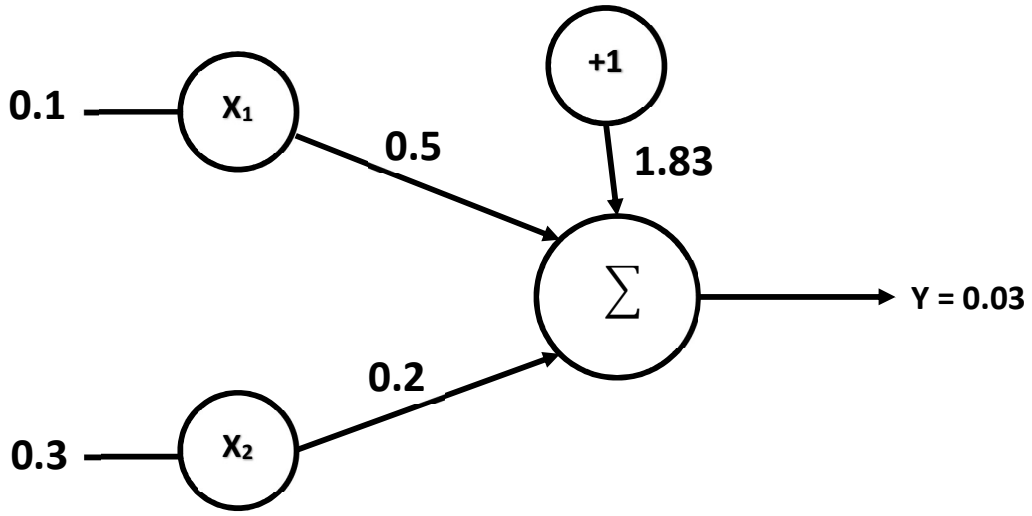
X = Current input at which the network made false prediction

 # How to Minimize Prediction Error?

There is a prediction error and it should be minimized until reaching an acceptable error.

There must be something to change in order to minimize the error. The only parameter to change is the weight. So, we can use the weight update equation.

Example:



Sol^n:

Trainning Data

X_1	X_2	Output
0.1	0.3	0.03

Initial Weights

W_1	W_2	b
0.5	0.2	1.83

Network Training

Steps to train our network:

1. Prepare activation function input (Sum of products between inputs and weights).
2. Activation function output.

 # Sum of Products:

$$s = b + \sum_{i=1}^{n} X_i * W_i$$

$$s = X_1 * W_1 + X_2 * W_2 + b$$

$$s = 0.1 * 0.5 + 0.3 * 0.2 + 1.83$$

$$s = 1.94$$

 # Activation Function:

Here, sigmoid activation function is used,

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

$$f(s) = \frac{1}{1+e^{-1.94}}$$

$$f(s) = \frac{1}{1+0.144}$$

$$f(s) = \frac{1}{1.144}$$

$$f(s) = 0.874 \quad \textbf{(Predicted Output)}$$

 # Prediction Error:

- After getting the predicated outputs, next is to measure the prediction error of the network.
- We can use the Square Error function as follows:

$$E = \frac{1}{2}(desired\ output - predicted\ output)^2$$

- Based on the predicted output, the prediction error is:

$$E = \frac{1}{2}(0.03 - 0.874)^2$$

$$E = \frac{1}{2}(-0.844)^2$$

$$E = \frac{1}{2}(0.713)$$

$$E = 0.357$$ *(The training process is updated till the sum of squared error is less than 0.001)*

 # Weight Training :

- We can use the weight update equation:

$$w_{new} = w_{old} + \propto [d - y] . X$$

Here,

w_{new} = New Updated Weights

w_{old} = Current Weights = **[1.83, 0.5, 0.2]** *[b, W_1, W_2]*

$\propto$ = Network Learning Rate or η (Eta) = 0.01

Assume any learning rate between 0 to 1 any positive value, default vlue is 0.1 or 0.01

$\boldsymbol{d}$ = Desired output = **0.03**

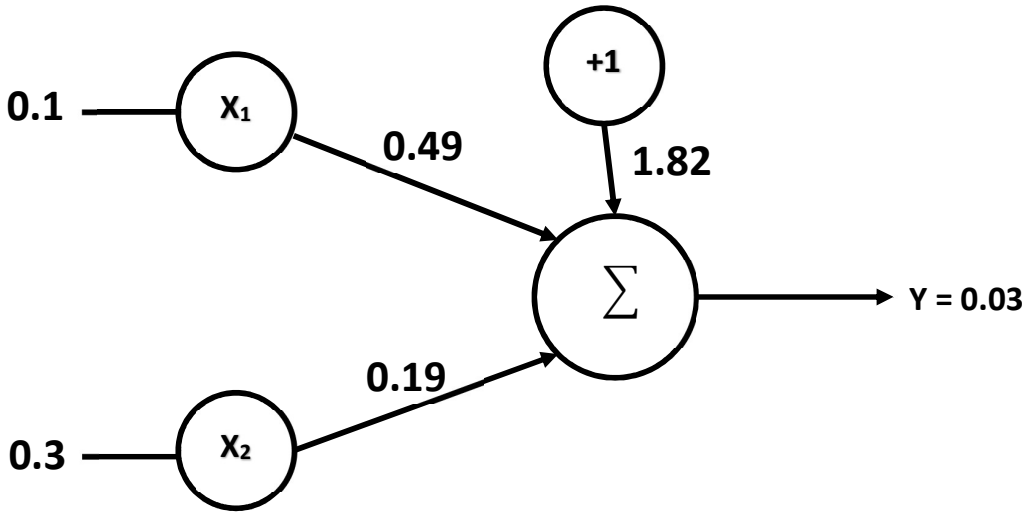
$\boldsymbol{y}$ = Predicted output = **0.874**

$\boldsymbol{X}$ = Current input at which the network made false prediction = **[+1, 0.1, 0.3]** *[b, X_1, X_2]*

$$\boldsymbol{w_{new} = w_{old} + \propto [d - y] . X}$$
$$= [\mathbf{1.83, 0.5, 0.2}] + \mathbf{0.01}[\mathbf{0.03 - 0.874}]. [\mathbf{+1, 0.1, 0.3}]$$
$$= [\mathbf{1.83, 0.5, 0.2}] + [\mathbf{-0.00844}]. [\mathbf{+1, 0.1, 0.3}]$$
$$= [\mathbf{1.83, 0.5, 0.2}] + [\mathbf{-0.0084, -0.0084, -0.0025}]$$
$$= [\mathbf{1.83, 0.5, 0.2}] - [\mathbf{0.0084, 0.0084, 0.0025}]$$
$$= [(\mathbf{1.83 - 0.0084}),\ (\mathbf{0.5 - 0.0084}),\ (\mathbf{0.2 - \ 0.0025})]$$
$$= [\mathbf{1.82, 0.49, 0.19}]$$ *[b, W_1, W_2]*

- **The new weights are:**

W_1	W_2	b
0.49	0.19	1.82



- Based on the new weights, the network will be re-trained.
- Continue these operations until prediction error reaches an acceptable value.
  i. Updating Weights
  ii. Re-training Networks
  iii. Calculating Prediction Error

6.1.5 Hopefield Network

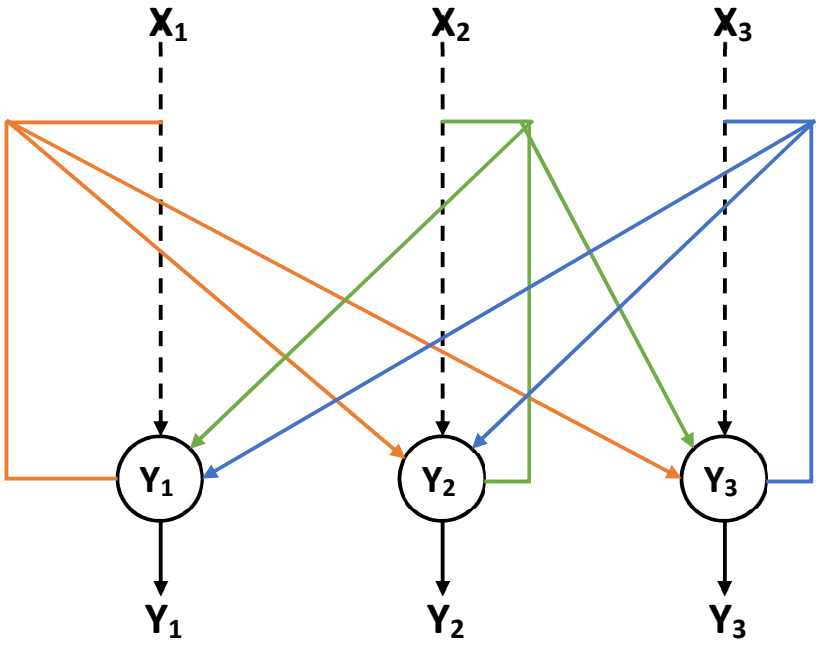
Hopfield neural network was invented by Dr. John J. Hopfield in 1982. It consists of a single layer which contains one or more fully connected recurrent neurons. The Hopfield network is commonly used for auto-association and optimization tasks.

Hopfield network which operates in a discrete line fashion or in other words, it can be said the input and output patterns are discrete vector, which can be either binary 0, 1 or bipolar +1, −1 in nature. The network has symmetrical weights with no self-connections i.e., $W_{ij} = W_{ji}$ and $W_{ii} = 0$.

Architecture

Following are some important points to keep in mind about discrete Hopfield network −

- This model consists of neurons with one inverting and one non-inverting output.
- The output of each neuron should be the input of other neurons but not the input of self.
- Weight/connection strength is represented by W_{ij}.
- Connections can be excitatory as well as inhibitory. It would be excitatory, if the output of the neuron is same as the input, otherwise inhibitory.
- Weights should be symmetrical, i.e. $W_{ij} = W_{ji}$



The output from **Y1** going to **Y2** and **Y3** have the weights **W12** and **W13** respectively.
Similarly, other arcs have the weights on them.

6.1.6 Kohonen Network

A Kohonen Network sometimes called **self-organizing map (SOM)** or **self-organizing feature map (SOFM)**.

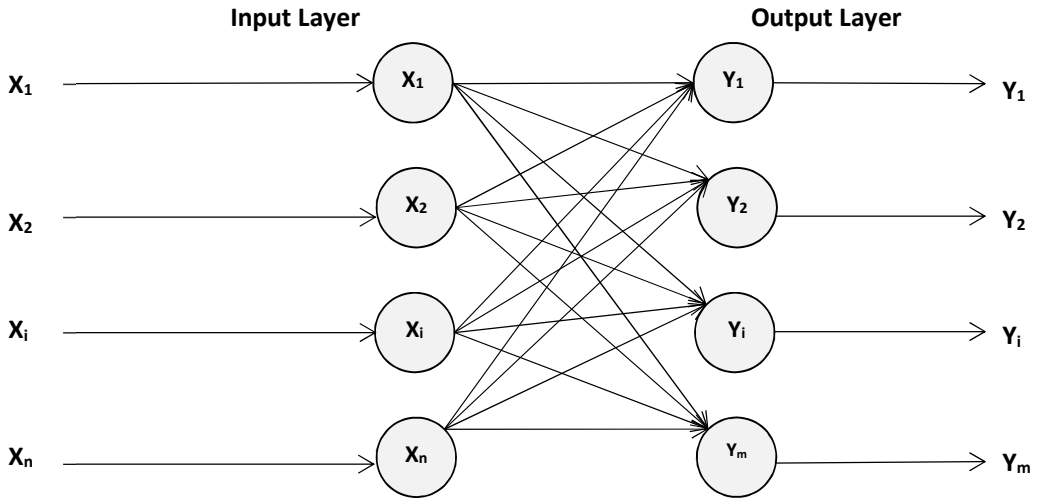
This network was introduced by the Finnish professor Teuvo Kohonen in the 1980s.

It is a type of artificial neural network (ANN) that is trained using unsupervised learning to produce a low-dimensional (typically two-dimensional), discretized representation of the input space of the training samples, called a map, and is therefore a method to do dimensionality reduction.

Kohonen's networks are one of basic types of self-organizing neural networks. The ability to self-organize provides new possibilities - adaptation to formerly unknown input data. It seems to be the most natural way of learning, which is used in our brains, where no patterns are defined. Those patterns take shape during the learning process, which is combined with normal work. Kohonen's networks are a synonym of whole group of nets which make use of self-organizing, competitive type learning method.

- The Kohonen model provides a topological mapping. It places a fixed number of input patterns from the input layer into higher dimensional output or Kohonen layer.
- Training in the Kohonen network begins with the winner's neighborhood of a fairly large size. Then, as training proceeds, the neighborhood size gradually decreases.

Architecture of Kohonen Network or SOM



Algorithm of Kohonen Network

Step 0: Initialize the weights W_{ij}. Random values may be assumed. Initialize the learning rate $(\boldsymbol{\eta}\ \boldsymbol{or}\ \boldsymbol{\alpha})$.

Step 1: While stopping condition false, do steps 2 to 8.

Step 2: For each input vector **X**, do steps 3 to 5.

Step 3: Calculate Square of the Euclidean Distance i.e. for each i=1 to m.

$$\boldsymbol{D(j) = \sqrt{\sum_{i=1}^{n}(W_{ij} - X_i)^2}}$$

You'll get same Winning Neuron from both the formula.

Or in some text book or in research papers you'll see

$$\boldsymbol{D(j) = \sum_{i=1}^{n}(W_{ij} - X_i)^2}$$

Don't be panic values are different but this will still yield the same output at last.

Step 4: The minimum $\boldsymbol{D(i,j)}$ is selected to be the winning unit.

Step 5: For all neurons j within a specific neighborhood of j and for all i, calculate new weights.

$$\boldsymbol{W_{ij}(new) = W_{ij}(old) + \alpha[X_i - W_{ij}(old)]}$$

Step 6: Update learning rate $\boldsymbol{\alpha}$. (Reduce the $\boldsymbol{\alpha}$ at each Iteration) $\boldsymbol{\alpha(t+1) = \frac{\alpha(t)}{2}}$

Step 7: Reduce radius of topological neighborhood at specified times.

Step 8: Test stopping condition. Typically, this is a small value of the learning rate with which the weight updates are insignificant.

Simple Example:

A Kohonen Self-Organizing Map (SOM) to cluster four vectors

Let the vector to be cluster be

(1, 1, 0, 0); (0, 0, 0, 1); (1, 0, 0, 0); (0, 0, 1, 1).

The maximum number of clusters to be formed is m=2.

Suppose the learning rate (geometric decrease) is $\boldsymbol{\alpha(0) = 0.6,}$

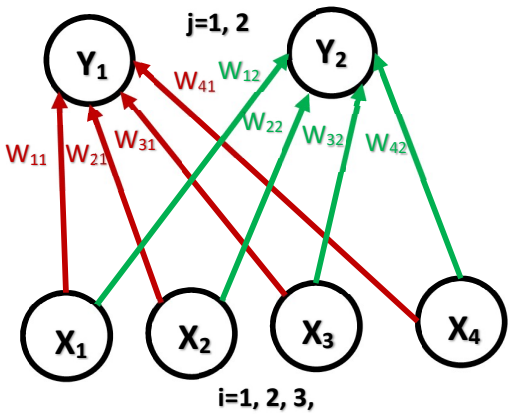
$$\boldsymbol{\alpha(t+1) = \frac{\alpha(t)}{2}}$$

$$\boldsymbol{or\ \ 0.5 * \alpha(t)}$$

Step 0: Let, Initial Weight Matrix between 0 to 1

$$\begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \\ W_{31} & W_{32} \\ W_{41} & W_{42} \end{bmatrix} = \begin{bmatrix} 0.2 & 0.8 \\ 0.6 & 0.4 \\ 0.5 & 0.7 \\ 0.9 & 0.3 \end{bmatrix}$$

Let, Initial learning rate: $\alpha(0) = 0.6$



Step 1: Begin Training.

Step 2: For the first vector, $(X_1, X_2, X_3, X_4) = (1, 1, 0, 0)$, do step 3 to 5.

Step 3:

$$D(1) = \sqrt{(X_1 - W_{11})^2 + (X_2 - W_{21})^2 + (X_3 - W_{31})^2 + (X_4 - W_{41})^2}$$

$$= \sqrt{(1 - 0.2)^2 + (1 - 0.6)^2 + (0 - 0.5)^2 + (0 - 0.9)^2}$$

$$= \sqrt{0.64 + 0.16 + 0.25 + 0.81}$$

$$= \sqrt{1.86}$$

$$= 1.36$$

$$D(2) = \sqrt{(X_1 - W_{12})^2 + (X_2 - W_{22})^2 + (X_3 - W_{32})^2 + (X_4 - W_{43})^2}$$

$$= \sqrt{(1 - 0.8)^2 + (1 - 0.4)^2 + (0 - 0.7)^2 + (0 - 0.3)^2}$$

$$= \sqrt{0.04 + 0.36 + 0.49 + 0.09}$$

$$= \sqrt{0.98}$$

$$= 0.98$$ **(Minimum Euclidean Distance)**

Step 4: The input vector is closest to output node 2, so j=2.

(As $D(2)$ has minimum Euclidean Distance)

Step 5: The weights on the winning unit/neuron are updated:

$$W_{ij}(new) = W_{ij}(old) + \alpha[X_i - W_{ij}(old)]$$

Where, learning rate $\alpha = 0.6$

$$W_{12}(new) = W_{12}(old) + 0.6[X_1 - W_{12}(old)]$$
$$W_{12}(new) = 0.8 + 0.6[1 - 0.8] = 0.92$$

$$W_{22}(new) = W_{22}(old) + 0.6[X_2 - W_{22}(old)]$$
$$W_{22}(new) = 0.4 + 0.6[1 - 0.4] = 0.76$$

$$W_{32}(new) = W_{32}(old) + 0.6[X_3 - W_{32}(old)]$$
$$W_{32}(new) = 0.7 + 0.6[0 - 0.7] = 0.28$$

$$W_{42}(new) = W_{42}(old) + 0.6[X_4 - W_{42}(old)]$$
$$W_{42}(new) = 0.3 + 0.6[0 - 0.3] = 0.12$$

The new weighted matrix is:

$$\begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \\ W_{31} & W_{32} \\ W_{41} & W_{42} \end{bmatrix} = \begin{bmatrix} 0.2 & 0.92 \\ 0.6 & 0.76 \\ 0.5 & 0.28 \\ 0.9 & 0.12 \end{bmatrix}$$

Step 2: For the second vector, $(X_1, X_2, X_3, X_4) = (0, 0, 0, 1)$, do step 3 to 5.

Step 3:

$$D(1) = \sqrt{(X_1 - W_{11})^2 + (X_2 - W_{21})^2 + (X_3 - W_{31})^2 + (X_4 - W_{41})^2}$$
$$= \sqrt{(0 - 0.2)^2 + (0 - 0.6)^2 + (0 - 0.5)^2 + (1 - 0.9)^2}$$
$$= \sqrt{0.04 + 0.36 + 0.25 + 0.01}$$
$$= \sqrt{0.66}$$
$$= 0.81 \text{ (Minimum Euclidean Distance)}$$

$$D(2) = \sqrt{(X_1 - W_{12})^2 + (X_2 - W_{22})^2 + (X_3 - W_{32})^2 + (X_4 - W_{43})^2}$$
$$= \sqrt{(0 - 0.92)^2 + (0 - 0.76)^2 + (0 - 0.28)^2 + (1 - 0.12)^2}$$
$$= \sqrt{0.85 + 0.58 + 0.08 + 0.77}$$
$$= \sqrt{2.28}$$
$$= 1.51$$

Step 4: The input vector is closest to output node 1, so j=1.

(As $D(1)$ has minimum Euclidean Distance)

Step 5: The weights on the winning unit/neuron are updated:

$$W_{ij}(new) = W_{ij}(old) + \alpha[X_i - W_{ij}(old)]$$

$Where, learning\ rate\ \boldsymbol{\alpha = 0.6}$

$$\boldsymbol{W_{11}(new) = W_{11}(old) + 0.6[X_1 - W_{11}(old)]}$$
$$\boldsymbol{W_{11}(new) = 0.2 + 0.6[0 - 0.2] = 0.08}$$

$$\boldsymbol{W_{21}(new) = W_{21}(old) + 0.6[X_2 - W_{21}(old)]}$$
$$\boldsymbol{W_{21}(new) = 0.6 + 0.6[0 - 0.6] = 0.06}$$

$$\boldsymbol{W_{31}(new) = W_{31}(old) + 0.6[X_3 - W_{31}(old)]}$$
$$\boldsymbol{W_{31}(new) = 0.5 + 0.6[0 - 0.5] = 0.20}$$

$$\boldsymbol{W_{41}(new) = W_{41}(old) + 0.6[X_4 - W_{41}(old)]}$$
$$\boldsymbol{W_{41}(new) = 0.9 + 0.6[1 - 0.9] = 0.96}$$

The new weighted matrix is:

$$\begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \\ W_{31} & W_{32} \\ W_{41} & W_{42} \end{bmatrix} = \begin{bmatrix} 0.08 & 0.92 \\ 0.06 & 0.76 \\ 0.20 & 0.28 \\ 0.96 & 0.12 \end{bmatrix}$$

Step 2: For the third vector, $(\boldsymbol{X_1, X_2, X_3, X_4}) = (\boldsymbol{1, 0, 0, 0})$, do step 3 to 5.

Step 3:

$$\boldsymbol{D(1) = \sqrt{(X_1 - W_{11})^2 + (X_2 - W_{21})^2 + (X_3 - W_{31})^2 + (X_4 - W_{41})^2}}$$
$$\boldsymbol{= \sqrt{(1 - 0.08)^2 + (0 - 0.06)^2 + (0 - 0.20)^2 + (0 - 0.96)^2}}$$
$$\boldsymbol{= \sqrt{0.8464 + 0.0036 + 0.04 + 0.9216}}$$
$$\boldsymbol{= \sqrt{1.8116}}$$
$$\boldsymbol{= 0.95}$$
$$\boldsymbol{D(2) = \sqrt{(X_1 - W_{12})^2 + (X_2 - W_{22})^2 + (X_3 - W_{32})^2 + (X_4 - W_{43})^2}}$$
$$\boldsymbol{= \sqrt{(1 - 0.92)^2 + (0 - 0.76)^2 + (0 - 0.28)^2 + (0 - 0.12)^2}}$$
$$\boldsymbol{= \sqrt{0.0064 + 0.58 + 0.08 + 0.0144}}$$
$$\boldsymbol{= \sqrt{0.6808}}$$
$$\boldsymbol{= 0.28}\ \textbf{(Minimum Euclidean Distance)}$$

Step 4: The input vector is closest to output node 2, so j=2.

(As $\boldsymbol{D(2)}$ has minimum Euclidean Distance)

Step 5: The weights on the winning unit/neuron are updated:

$$\boldsymbol{W_{ij}(new) = W_{ij}(old) + \alpha[X_i - W_{ij}(old)]}$$

$Where, learning\ rate\ \boldsymbol{\alpha = 0.6}$

$\boldsymbol{W_{12}(new) = W_{12}(old) + 0.6[X_1 - W_{12}(old)]}$
$\boldsymbol{W_{12}(new) = 0.92 + 0.6[1 - 0.92] = 0.968}$

$\boldsymbol{W_{22}(new) = W_{22}(old) + 0.6[X_2 - W_{22}(old)]}$
$\boldsymbol{W_{22}(new) = 0.76 + 0.6[0 - 0.76] = 0.304}$

$\boldsymbol{W_{32}(new) = W_{32}(old) + 0.6[X_3 - W_{32}(old)]}$
$\boldsymbol{W_{32}(new) = 0.28 + 0.6[0 - 0.28] = 0.112}$

$\boldsymbol{W_{42}(new) = W_{42}(old) + 0.6[X_4 - W_{42}(old)]}$
$\boldsymbol{W_{42}(new) = 0.12 + 0.6[0 - 0.12] = 0.048}$

The new weighted matrix is:

$$\begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \\ W_{31} & W_{32} \\ W_{41} & W_{42} \end{bmatrix} = \begin{bmatrix} 0.08 & 0.968 \\ 0.06 & 0.304 \\ 0.20 & 0.112 \\ 0.96 & 0.048 \end{bmatrix}$$

Step 2: For the fourth vector, $(X_1, X_2, X_3, X_4) = (0, 0, 1, 1)$, do step 3 to 5.

Step 3:

$\boldsymbol{D(1) = \sqrt{(X_1 - W_{11})^2 + (X_2 - W_{21})^2 + (X_3 - W_{31})^2 + (X_4 - W_{41})^2}}$

$\boldsymbol{= \sqrt{(0 - 0.08)^2 + (0 - 0.06)^2 + (1 - 0.20)^2 + (1 - 0.96)^2}}$

$\boldsymbol{= \sqrt{0.0064 + 0.0036 + 0.64 + 0.0016}}$

$\boldsymbol{= \sqrt{0.6516}}$

$\boldsymbol{= 0.2}$ **(Minimum Euclidean Distance)**

$\boldsymbol{D(2) = \sqrt{(X_1 - W_{12})^2 + (X_2 - W_{22})^2 + (X_3 - W_{32})^2 + (X_4 - W_{43})^2}}$

$\boldsymbol{= \sqrt{(0 - 0.968)^2 + (0 - 0.304)^2 + (1 - 0.112)^2 + (1 - 0.048)^2}}$

$\boldsymbol{= \sqrt{0.937024 + 0.092416 + 0.788544 + 0.906304}}$

$\boldsymbol{= \sqrt{2.726032}}$

$\boldsymbol{= 0.9757}$

Step 4: The input vector is closest to output node 1, so j=1.

(As $D(1)$ has minimum Euclidean Distance)

Step 5: The weights on the winning unit/neuron are updated:

$$\boldsymbol{W_{ij}(new) = W_{ij}(old) + \alpha[X_i - W_{ij}(old)]}$$

$Where, learning\ rate\ \boldsymbol{\alpha = 0.6}$

$$\boldsymbol{W_{11}(new) = W_{11}(old) + 0.6[X_1 - W_{11}(old)]}$$
$$\boldsymbol{W_{11}(new) = 0.08 + 0.6[0 - 0.08] = 0.032}$$

$$\boldsymbol{W_{21}(new) = W_{21}(old) + 0.6[X_2 - W_{21}(old)]}$$
$$\boldsymbol{W_{21}(new) = 0.24 + 0.6[0 - 0.24] = 0.096}$$

$$\boldsymbol{W_{31}(new) = W_{31}(old) + 0.6[X_3 - W_{31}(old)]}$$
$$\boldsymbol{W_{31}(new) = 0.20 + 0.6[1 - 0.20] = 0.680}$$

$$\boldsymbol{W_{41}(new) = W_{41}(old) + 0.6[X_4 - W_{41}(old)]}$$
$$\boldsymbol{W_{41}(new) = 0.96 + 0.6[1 - 0.96] = 0.984}$$

The new weighted matrix is:

$$\begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \\ W_{31} & W_{32} \\ W_{41} & W_{42} \end{bmatrix} = \begin{bmatrix} 0.032 & 0.968 \\ 0.096 & 0.304 \\ 0.680 & 0.112 \\ 0.984 & 0.048 \end{bmatrix}$$

As all the input vectors are done then, Iteration 1 is completed.

Step 6: Reduce the learning rate:

$$\boldsymbol{\alpha(t+1) = \frac{\alpha(t)}{2}\ \ or\ \ 0.5 * \alpha(t)}$$

So, $\boldsymbol{\frac{0.6}{2} = 0.3}$

Step1: While stopping condition false, do steps 2 to 8.

Now, having a new learning rate 0.3

The weight matrix after the second epoch/iteration of training is:

$$\begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \\ W_{31} & W_{32} \\ W_{41} & W_{42} \end{bmatrix} = \begin{bmatrix} 0.015 & 0.9776 \\ 0.047 & 0.5128 \\ 0.633 & 0.0776 \\ 0.992 & 0.0336 \end{bmatrix}$$

Modifying the adjustment procedure for the learning rate α so that it decreases geometrically from 0.6 to 0.01 over 100 epochs/iterations gives the more precise output. Although the Neural network Kohonen SOM algorithm automatically stops when it gets the more accurate result.

But for the exam point of view you can just do up to 1st Iteration. Not necessary to do all the iteration to get result. It'll be time consuming.

At Iteration 100: Weight matrix:

$$\begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \\ W_{31} & W_{32} \\ W_{41} & W_{42} \end{bmatrix} = \begin{bmatrix} 0.0 & 1.0 \\ 0.0 & 0.5 \\ 0.5 & 0.0 \\ 1.0 & 0.0 \end{bmatrix}$$

Cluster 1 ← → Cluster 2

The first column of which is the average of the two vectors placed in cluster 1 and the second column of which is the average of the two vectors placed in cluster 2.

TEST NETWORK

Suppose the input pattern is 1100.

Then,

$$D(j) = \sqrt{(X_1 - W_{1j})^2 + (X_2 - W_{2j})^2 + (X_3 - W_{3j})^2 + (X_4 - W_{4j})^2}$$

$$D(1) = \sqrt{(1-0)^2 + (1-0)^2 + (0-0.5)^2 + (0-1)^2} = 3.25$$

$$D(2) = \sqrt{(1-1)^2 + (1-0.5)^2 + (0-0)^2 + (0-0)^2} = 0.25$$

Thus **Neuron 2** is the **"Winner"**, and is the localized active region of the SOM. Notice that we may label this input pattern to belong to **cluster 2**.

For all the other patterns, we find the cluster are as listed below.

X_1	X_2	X_3	X_4	Cluster
1	1	0	0	**2**
0	0	0	1	**1**
1	0	0	0	**2**
0	0	1	1	**1**

Why Use Kohonen Network or SOM (Self-Organizing Maps)?

- Find clusters in large amounts of data (dimensionality reduction).
- Combine diverse datasets to find patterns.
- Powerful Visualization.
- Application in any field.

6.2 Expert System

Expert System are computer programs built for commercial application using the programming techniques of AI which are developed for problem solving.

Built for varieties of purposes including **Medical Diagnosis, Electronic Fault Finding, Mineral Prospecting, Computer System Configuration,** etc.

Definition of Expert System:

An Expert System is a collection of programs or Computer Software that solves problems in the domain of interest. *It is called system because it consists of both problem-solving component and a support component.*

The process of building Expert System is called Knowledge Engineering and is done by Knowledge Engineer.

What is an Expert System?

An expert system is a computer program that is designed to solve complex problems and to provide decision-making ability like a human expert. It performs this by extracting knowledge from its knowledge base using the reasoning and inference rules according to the user queries.

The expert system is a part of AI, and the first Expert System was developed in the year 1970, which was the first successful approach of artificial intelligence. It solves the most complex issue as an expert by extracting the knowledge stored in its knowledge base. The system helps in decision making for complex problems using both facts and heuristics like a human expert. It is called so because it contains the expert knowledge of a specific domain and can solve any complex problem of that particular domain. These systems are designed for a specific domain, such as medicine, science, etc.

The performance of an expert system is based on the expert's knowledge stored in its Knowledge Base (KB). The more knowledge stored in the KB, the more that system improves its performance. One of the common examples of an Expert System is a **suggestion of spelling errors** while typing in the Google search box.

Knowledge Engineer:

- It is a human with a background of computer and AI who knows how to build an Expert System.
- Decides how to represent the knowledge from human expert or other sources.
- Creates a relation with human expert to elicit knowledge from him/her.

Knowledge Engineering:

- It is the acquisition of knowledge and gives report to knowledge engineer.

Expert:

- Evaluates the expert system and gives report to knowledge engineer.
- The process continues until the performance is found satisfactory by the experts.

Expert system provides the following important features:

- Facility for non-expert personnel to solve problems that require some expertise
- Speedy solution.
- Reliable solution.
- Cost reduction.
- Power to manage without human expert.
- Wider areas of knowledge.

Expert system provides the following important features:

- Human experts are difficult to find.
- Human experts are expensive.
- Knowledge improvement is important.
- The available information is poor, partial, incomplete.
- Problems are incompletely defined.
- There is lack of knowledge among all those who need it.
- The problem is rapidly changing legal rules and codes.

Some popular examples of the Expert System are as follows:

DENDRAL: It was an artificial intelligence project that was made as a chemical analysis expert system. It was used in organic chemistry to detect unknown organic molecules with the help of their mass spectra and knowledge base of chemistry.

MYCIN: It was one of the earliest backward chaining expert systems that was designed to find the bacteria causing infections like bacteremia and meningitis. It was also used for the recommendation of antibiotics and the diagnosis of blood clotting diseases.

PXDES: It is an expert system that is used to determine the type and level of lung cancer. To determine the disease, it takes a picture from the upper body, which looks like the shadow. This shadow identifies the type and degree of harm.

CaDeT: The CaDeT expert system is a diagnostic support system that can detect cancer at early stages.

Characteristics of Expert System:

- **High Performance:**
  - The expert system provides high performance for solving any type of complex problem of a specific domain with high efficiency and accuracy.

- **The Highest Level of Expertise:**
  - The expert system offers the highest level of expertise. It provides efficiency, accuracy and imaginative problem-solving.

- **Adequate response time:**
  - Should respond within reasonable amount of time comparable to or better than time taken by human experts to reach the decision.
  - An Expert System interacts in a very reasonable period of time with the user.
  - The total time must be less than the time taken by an expert to get the most accurate solution for the same problem.

- **Good reliability:**
  - The Expert System needs to be reliable, and it must not make any mistake.

- **Flexibility:**
  - Huge Knowledge → Must have efficient mechanism to add, change and delete the knowledge.

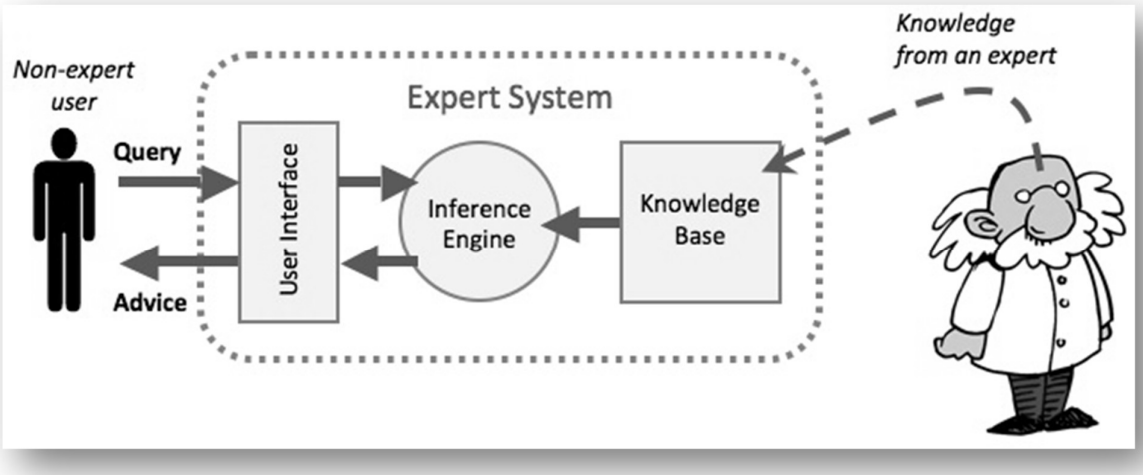
- **Highly responsive:**
  - Expert System provides the result for any complex query within a very short period of time.

- **Capable of handling challenging decision & problems:**
  - An expert system is capable of handling challenging decision problems and delivering solutions.

Components of Expert System:

An expert system mainly consists of three components:

1. User Interface
2. Inference Engine
3. Knowledge Base



1. **User Interface:**

- User interface provides interaction between user of the Expert System and Expert System itself. It is generally Natural Language Processing so as to be used by the user who is well-versed in the task domain. The user of the Expert System need not be necessarily an expert in Artificial Intelligence.
- It explains how Expert System has arrived at a particular recommendation
- With the help of a user interface, the expert system interacts with the user, takes queries as an input in a readable format, and passes it to the inference engine. After getting the response from the inference engine, it displays the output to the user. In other words, it is an interface that helps a non-expert user to communicate with the expert system to find a solution.

2. **Inference Engine (Rules of Engine):**

- The inference engine is known as the brain of the expert system as it is the main processing unit of the system. It applies inference rules to the knowledge base to derive a conclusion or deduce new information. It helps in deriving an error-free solution of queries asked by the user.
- With the help of an inference engine, the system extracts the knowledge from the knowledge base.
- **There are two types of inference engine:**

  i. **Deterministic Inference Engine:** The conclusions drawn from this type of inference engine are assumed to be true. It is based on facts and rules.

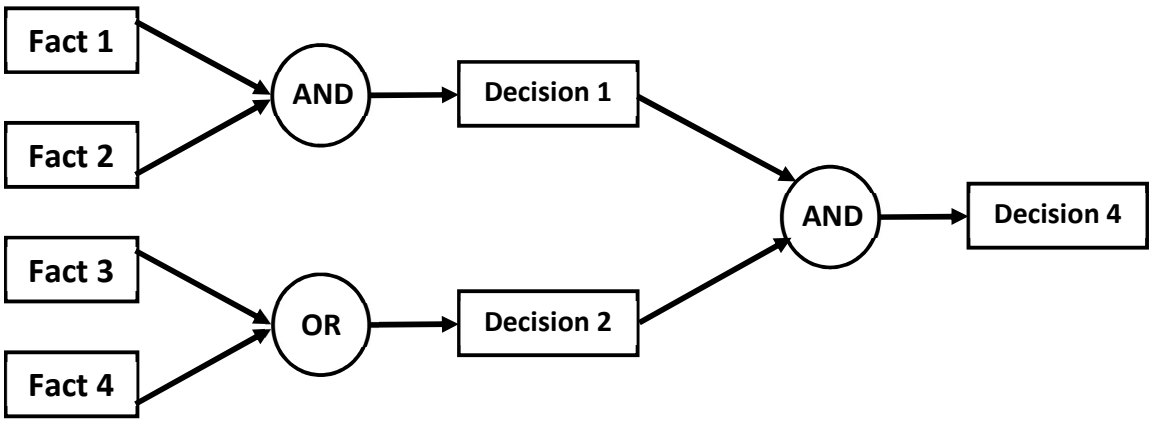
ii. **Probabilistic Inference engine:** This type of inference engine contains uncertainty in conclusions, and based on the probability.

Inference engine uses the below modes to derive the solutions:

Forward Chaining: It starts from the known facts and rules, and applies the inference rules to add their conclusion to the known facts.

Backward Chaining: It is a backward reasoning method that starts from the goal and works backward to prove the known facts.

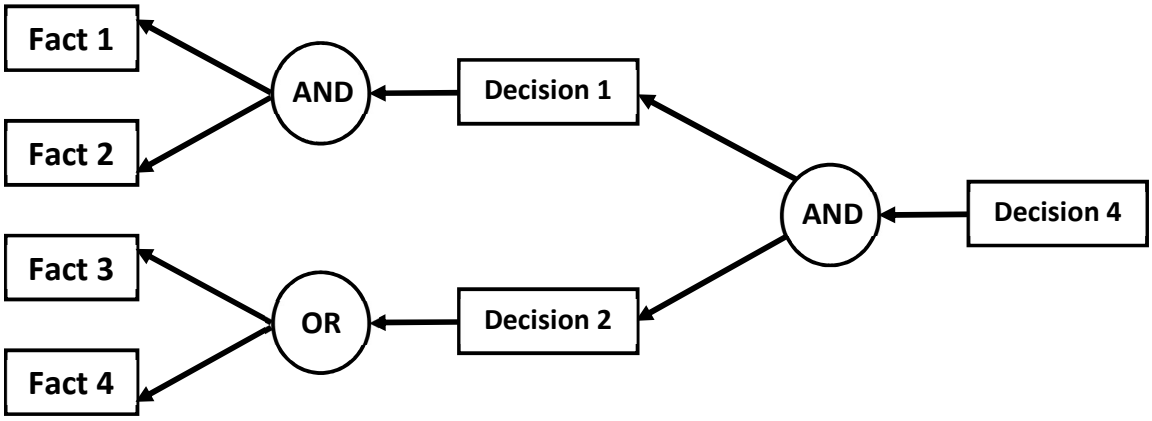
Forward Chaining:



- The forward chaining (or forward reasoning) is one of the two main methods of reasoning when using an inference engine.
- It is a strategy of an expert system to answer the question, **"What can happen next?"**
- It is also known as **data driven** inference technique.
- Forward chaining matches the set of conditions and inform results from these conditions. Basically, forward chaining starts from a new data and aims for any conclusion.
- It is **bottom-up** reasoning.
- It uses a **breadth first search** technique.
- It continues until no more rules can be applied or some cycle limit is met.
- Here, the Inference Engine follows the chain of conditions and derivations and finally deduces the outcome. It considers all the facts and rules, and sorts them before concluding to a solution.

- This strategy is followed for working on conclusion, result, or effect. For example, **prediction of share market status** as an effect of changes in interest rates.
- It is mostly used in commercial applications i.e. event driven systems are common example of forward chaining.
- It can create an infinite number of possible conclusions.
- **For example:** Let say, **if it is cold then I will wear a sweater**. Here **"it is cold"** is the ***data*** and **"I will wear a sweater"** is a ***decision***. It was already known that it is cold that is why it was decided to wear a sweater, this process is called **forward chaining**.

Backward Chaining:



- The forward chaining (or backward reasoning) is an interface method described colloquially *(बोलचाल) (Not formal or literary)* as working backward from goal.
- It is used in automated theorem provers, interface engines, proof assistants, and other artificial intelligence applications.
- With this strategy, an expert system finds out the answer to the question, **"Why this happened?"**
- It is also called as **goal driven** information technique.
- It is a backward search from goal to the conditions used to get the goal. Basically, it starts from the possible conclusion or goal and aims for necessary data.
- It is **top-down** reasoning.
- It uses **depth first search** technique.
- It processes operations in a backward direction from end to start, it will stop when matching initial condition is met.

- On the basis of what has already happened, the Inference Engine tries to find out which conditions could have happened in the past for this result.
- This strategy is followed for finding out cause or reason. For example, **diagnosis of blood cancer in humans**.
- It is used in interrogative commercial applications i.e. finding items that fulfill possible goals.
- Number of possible final answers is reasonable.
- **For example:** Let say, **if it is cold then I will wear a sweater**. Here, we have our possible conclusion **"I will wear a sweater".** If I am wearing a sweater then it can be started that "**it is cold**" that is why I am wearing a sweater. Hence it was derived in backward direction, this process is called **forward chaining**.

3. Knowledge Base:

- The knowledgebase is a type of storage that stores knowledge acquired from the different experts of the particular domain.
- It is considered as big storage of knowledge. The more the knowledge base, the more precise will be the Expert System.
- It is similar to a database that contains information and rules of a particular domain or subject.
- One can also view the knowledge base as collections of objects and their attributes. Such as a Lion is an object and its attributes are it is a mammal, it is not a domestic animal, etc.

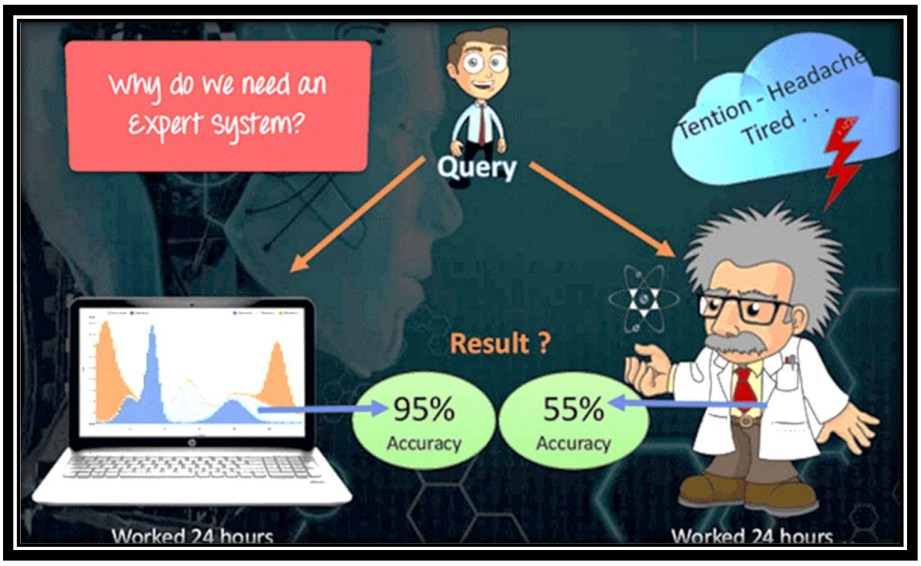
Components of Knowledge Base

Factual Knowledge: The knowledge which is based on facts and accepted by knowledge engineers comes under factual knowledge.

Heuristic Knowledge: This knowledge is based on practice, the ability to guess, evaluation, and experiences.

Knowledge Representation: It is used to formalize the knowledge stored in the knowledge base using the If-else rules.

Knowledge Acquisitions: It is the process of extracting, organizing, and structuring the domain knowledge, specifying the rules to acquire the knowledge from various experts, and store that knowledge into the knowledge base.



Advantages of Expert System

- Increased output and productivity.
- Decreased decision-making time.
- Increased process and product quality.
- Reduced downtime.
- Capture scarce expertise.
- Flexibility.
- Easier equipment operation.
- Elimination of expensive equipment.
- Operation in hazardous environment.
- Accessibility to knowledge and help desks.
- Integration of several experts' opinions.
- Can work with incomplete or uncertain information.
- Provide training.
- Enhancement of problem solving and decision making.
- Improved decision-making processes.
- Improved decision quality.
- Ability to solve complex problems.
- Knowledge transfer to remote locations.
- Enhancement of other MIS.

Limitations of Expert System

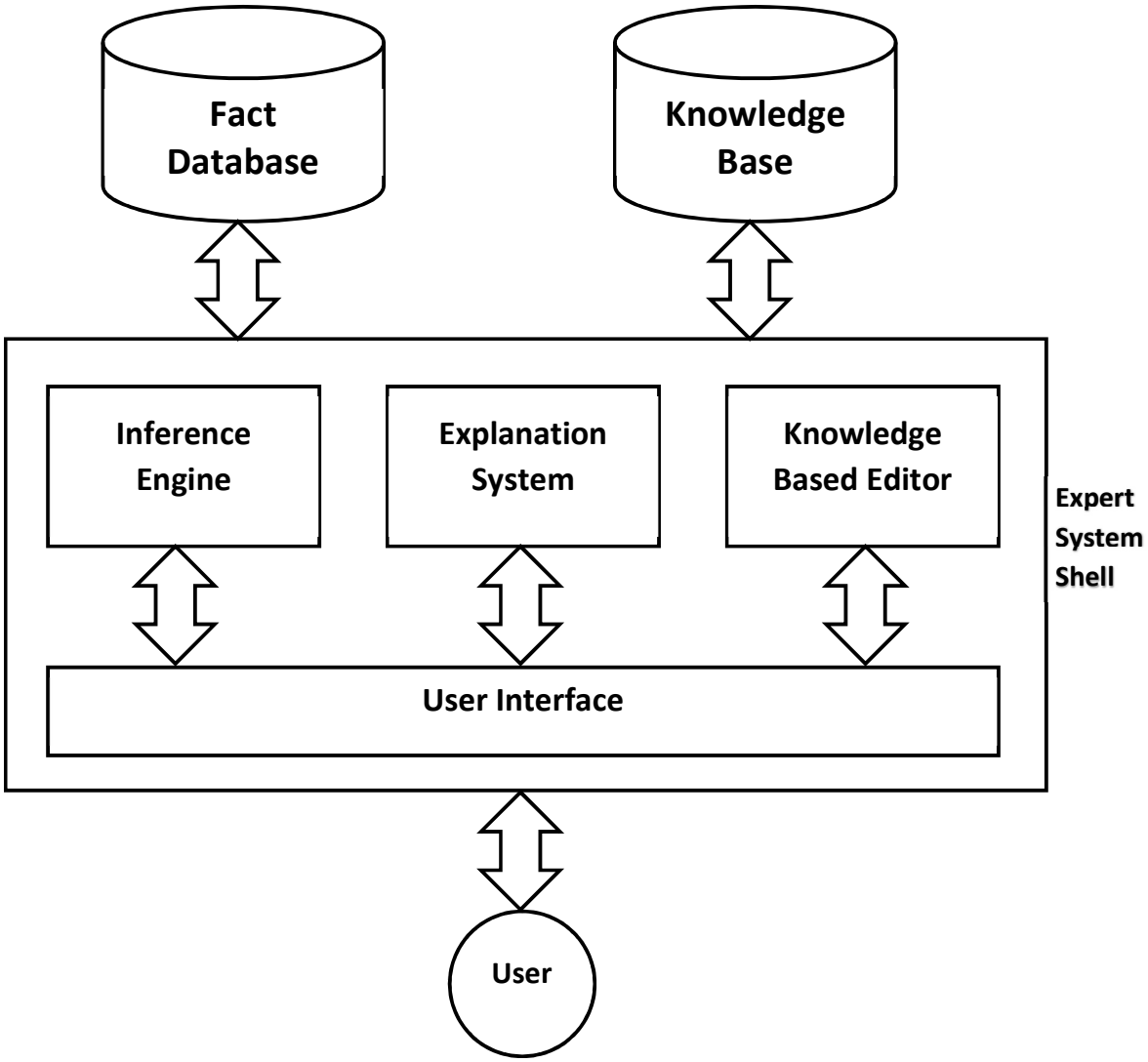
- Knowledge is not always readily available.
- Expertise can be hard to extract from humans.
- Each expert's approach may be different yet correct.
- Hard, even for a highly skilled expert, to work under time pressure.
- Expert System users have natural cognitive limits.
- Expert System works well only in narrow domain of knowledge.
- Most experts have no independent means to validate their conclusions.
- Experts' vocabulary often limited and highly technical.
- Knowledge engineers are rare and expensive.
- Lack of trust by end-users.
- Knowledge transfer subject to a host of perceptual and judgmental biases.
- Expert System may not be able to arrive at valid conclusions.
- Expert System sometimes produce incorrect recommendations.

Applications of Expert Systems and Artificial Intelligence:

- Credit granting
- Information management and retrieval
- AI and expert systems embedded in products
- Hospitals and medical facilities
- Help desks and assistance
- Employee performance evaluation
- Loan analysis
- Virus detection
- Repair and maintenance
- Shipping
- Marketing
- Warehouse optimization
- Plant layout

6.2.1 Architecture of an Expert System

The Architecture of Expert System



- Expert knowledge derived from human experts.
- **Purpose:**
  - Diagnose Illnesses
  - Provide Recommendations
  - Solve other complex problems

- **Knowledge Base:** Database of rules (Domain Knowledge).
- **Explanation System:** Explains the decisions the system makes.
- **User Interface:** The means by which the user interacts with the expert system.
- **Knowledge Base Editor:** Allows the user to edit the information in the knowledge base.
- **Expert System Shell:** The part of an expert system that does not contain any domain specific or case specific knowledge is the expert system shell. A single expert system shell can be used to build a number of different expert systems. An example of an expert system shell is CLIPS. CLIPS is a C language Integrated Production System – an expert system shell. CLIPS use a LIPS-like notation to enter rules.

(Refer from component of Expert System which is discussed above)

A Simple Example of Medical Expert System:

- **Rules**
  1. If headache then prescribe pain killer.
  2. If headache and sore throat and coughing than diagnose flu.
  3. If tired and headache then diagnose glandular fever.
  4. If tired and sore throat then diagnose tonsillitis.
  5. If tired than diagnose stress.

- **User Interaction**
  - **Patient:** I have a headache
  - **System:** I suggest you take a pain killer. Do you have a sore throat?
  - **Patient:** Yes, I do.
  - **System:** Have you been coughing?
  - **Patient:** No, why do you ask?
  - **System:** If you had a head ache and a sore throat and had been coughing, I would have to conclude you might have the flu. It doesn't sound as though you have the flu? Have you been feeling tired recently?
  - **Patient:** Yes, I have.
  - **System:** It sounds as though you may have glandular fever or tonsillitis.

6.2.2 Development of Expert System

Development of Expert Systems: General Steps

The process of Expert System development is iterative. Steps in developing the ES include −

Identify Problem Domain

- The problem must be suitable for an Expert System to solve it.
- Find the experts in task domain for the Expert System project.
- Establish cost-effectiveness of the system.

Design the System

- Identify the Expert System Technology
- Know and establish the degree of integration with the other systems and databases.
- Realize how the concepts can represent the domain knowledge best.

Develop the Prototype

From Knowledge Base: The knowledge engineer works to −

- Acquire domain knowledge from the expert.
- Represent it in the form of If-THEN-ELSE rules.

Test and Refine the Prototype

- The knowledge engineer uses sample cases to test the prototype for any deficiencies in performance.
- End users test the prototypes of the Expert System.

Develop and Complete the Expert System

- Test and ensure the interaction of the ES with all elements of its environment, including end users, databases, and other information systems.
- Document the Expert System project well.
- Train the user to use Expert System.

Maintain the System

- Keep the knowledge base up-to-date by regular review and update.
- Cater for new interfaces with other information systems, as those systems evolve.

Roles of Individuals who interact with the system: or
Participants in Expert System Development and Use:

- **Domain Expert:** The individual who currently are expert in solving the problems; here the system is intended to solve.

- **Knowledge Engineer:** The individual who encodes the expert's knowledge in a declarative form that can be used by the expert system.

- **End-User:** The individual who will be consulting with the system to get advice which would have been provided by the expert.

- **System Engineer:** Builds the user interface, designs the declarative format of the knowledge base, and implements the inference engine.

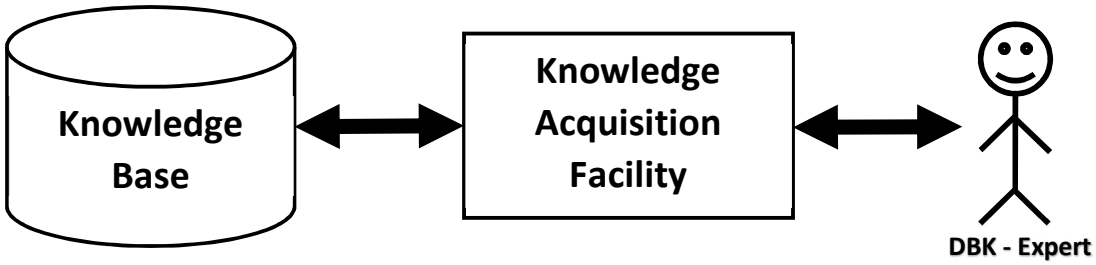
Other Key terms used in Expert systems

 # Facts and Rules:

- A fact is a small portion of important information. Facts on their own are of very limited use.
- The rules are essential to select and apply facts to a user problem.

 # Knowledge Acquisition:

- The term knowledge acquisition means how to get required domain knowledge by the expert system.
- The entire process starts by extracting knowledge from a human expert, converting the acquired knowledge into rules and injecting the developed rules into the knowledge base.
- Knowledge acquisition covers all forms of knowledge and any methods by which they may be obtained.
- It provides a convenient and efficient means of capturing and storing all components of the knowledge base.



6.3 Natural Language Processing

Introduction of NLP:

Natural Language Processing (NLP) refers to AI method of communicating with an intelligent system using a natural language such as English.

Processing of Natural Language is required when you want an intelligent system like robot to perform as per your instructions, when you want to hear decision from a dialogue based clinical expert system, etc.

The field of NLP involves making computers to perform useful tasks with the natural languages humans use. The input and output of an NLP system can be −

- Speech
- Written Text

Components of NLP:

There are two components of NLP as given

1. **Natural Language Understanding (NLU):**

- Understanding involves the following tasks.
- Mapping the given input in natural language into useful representations.
- Analyzing different aspects of the language.

2. **Natural Language Generation (NLG):**

- It is the process of producing meaningful phrases and sentences in the form of natural language from some internal representation.

It involves:

- Text planning − It includes retrieving the relevant content from knowledge base.
- Sentence planning − It includes choosing required words, forming meaningful phrases, setting tone of the sentence.
- Text Realization − It is mapping sentence plan into sentence structure.

The NLU is harder than NLG.

Difficulties in NLU / Ambiguities in Natural Language Processing

NL has an extremely rich form and structure. It is very ambiguous. There can be different levels of ambiguity:

- **Lexical ambiguity:** It is at very primitive level such as word-level. This type of ambiguity represents words that can have multiple assertions.

  For example, treating the word **"board"** as noun or verb? Another example the word "back" can be a noun (back stage), an adjective (back door) or an adverb (back away).

- **Syntax Level ambiguity:** A sentence can be parsed in different ways. Syntactic ambiguity is not usually something one strives for in clear communication, however, it does have its uses. One of the most entertaining is when double meanings are applied for comedic purposes. Ignoring the accepted context of a phrase and embracing an alternative meaning often ends in a laugh. Each of these general examples of ambiguity can carry double meanings:

  For example, **"He lifted the beetle with red cap."** *(उसले रातो टोपीले गोब्रे कीरा उठायो।)*

  − **Did he use red cap to lift the beetle or he lifted a beetle that had red cap?**

  − *के उसले गोब्रे कीरा उठाउन रातो टोपी प्रयोग गर्यो कि उसले रातो टोपीमा भएको गोब्रे कीरा उठायो?*

  "**I saw someone on the hill with a telescope."** *(मैले टेलिस्कोपमा कसैलाई देखें।)*

  − **Did you use a telescope to see someone on the hill or did you see someone on the hill holding a telescope?**

  − *तपाईंले पहाडमा कसैलाई हेर्न टेलिस्कोप प्रयोग गर्नुभयो वा पहाडमा कसैलाई टेलिस्कोप समातेको तपाईंले देख्नुभयो?*

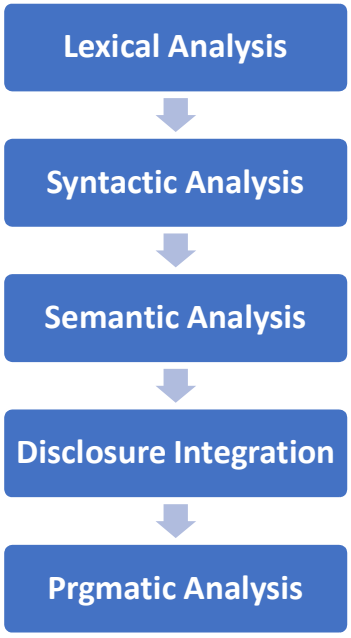
- **Referential ambiguity:** Referring to something using pronouns. For example, Rima went to Gauri. She said, **"I am tired."** − Exactly who is tired?

- One input can mean different meanings.

- Many inputs can mean the same thing.

Steps in NLP

There are generally five steps in Natural Language Processing:



1. **Lexical Analysis**: We have to analyze the structure of words. The collection of words and phrases in a language is a lexicon of a language. It involves identifying and analyzing the structure of words. Lexicon of a language means the collection of words and phrases in a language. Lexical analysis is dividing the whole chunk of text into paragraphs, sentences, and words.

2. **Syntactic Analysis (Parsing)**: It involves analysis of words in the sentence for grammar and arranging words in a manner. That shows the relationship among the words. The sentence such as **"The school goes to boy" or "I eats rice"** is rejected by English syntactic analyzer.

3. **Semantic Analysis:** It draws the exact meaning or the dictionary meaning from the text. The text is checked for meaningfulness. It is done by mapping syntactic structures and objects in the task domain. The semantic analyzer disregards sentence such as **"hot ice-cream" or "I eat house"**.

4. **Discourse Integration:** In this step, the meaning of any sentence depends upon the meaning of the previous sentence. In addition, it also brings about the meaning of immediately succeeding sentence. It is a sense of the context. The meaning of any single sentence depends upon the sentences that precedes it and also invokes the meaning of the sentences that follow it. Example: the word **"He"** and **"it"** in the sentence **"He likes it"** depends upon the prior discourse context. Means, who is he and what he likes is in the previous sentence. *("Kaji bought an AI book. He likes it.")* – Full Sentence

5. **Pragmatic Analysis:** In this step, data is interpreted on what it actually meant. It involves deriving those aspects of language which require real world knowledge. It is a study of language use. Relation between context and meaning. Pragmatics are the study of words and their meaning in a language with concern to their context. Additionally, focuses on the meaning of words according to the context and their inferred meanings as well. Studies the intended or the inferred meaning as well. Example: **"Close the window?", "Do you know what time it is?"**

   Conversations between A and B:
   A: I have a 14 years old son.
   B: Well that's all right.
   A: I also have a dog.
   B: Oh, I'm Sorry!

Syntactic Processing

- Syntactic Processing is the step in which a flat input sentence is converted into a hierarchical structure that corresponds to the units of meaning in the sentence.
- This process is called parsing.
- It plays an important role in natural language understanding systems for two reasons:
    - Semantic processing must operate on sentence constituent. If there is no syntactic parsing step, then the semantics system must decide on its own constituents. If parsing is done, on the other hand, it constrains the number of constituents that semantics can consider.

  - Syntactic parsing is computationally less expensive than is semantic processing. Thus, it can play a significant role in reducing overall system complexity.
- Although it is often possible to extract the meaning of a sentence without using grammatical facts, it is not always possible to do so. Consider the examples:
  - The satellite orbited Mars.
  - Mars orbited the satellite.

- Almost all the systems that are actually used have two main components:
  - **A declarative representation, called a grammar**, of the syntactic facts about the language.
  - **A procedure, called parser**, that compares the grammar against input sentences to produce parsed structures.

Grammars and Parsers

✓ In Context of NLP, Parsing implies analyzing a sentence syntactically to assign syntactic tags (subject, verb, object etc.) to provide constituent structure (noun phrase, verb phrase etc.) or to characterize the syntactic relations between two words.

✓ Parsing technique is further divided into **rule-based parsing** and **statistical parsing**.

Rule-based parsing:

- In rule-based parsing, syntactic structure of language is provided in form of linguistics rules which can be coded as production rules that are similar to context free rules.
- Production rules are defined using non-terminal and terminal symbols.

Statistical parsing:

- Require large corpora and linguistic knowledge is represented as statistical parameters or probability.

✓ Once the grammar rules are defined, a sentence is parsed using the grammar and tree kind of structure is built, if sentence is syntactically correct. This tree is called parse tree.

✓ Parsing can be done in two methods: **top-down parsing** and **bottom up parsing**.

i. **Bottom-up parsing:** we start with the words in the sentence and apply grammar rules in the backward direction until a single tree is produced whose root matches with start symbol.

ii. **Top-down parsing:** we start with start symbol and apply grammar rules in forward direction until the terminal symbol of the parse tree corresponds to the word in the sentence.

- ✓ The choice between these two approaches is similar to the choice between forward and backward reasoning in other problem-solving tasks.
- ✓ The most important consideration is the branching factor. Is it greater going backward or forward?
- ✓ Sometimes these two approaches are combined to a single method called "bottom-up parsing with top-down filtering".

Consider the simple context free grammar for English language.

Rules	Dictionary Words
$< S > \quad -> \quad < NP >< VP >$	$< Det > \quad -> a \mid an \mid the \mid this$
$< NP > \quad -> < Adj >< N >$	$< N > \quad -> girl \mid apple \mid john$
$< NP > \quad -> < Det >< N >$	$< Adj > \quad -> cute \mid smart$
$< NP > \quad -> < Det >< Adj >< N >$	$< V > \quad -> sings \mid eats \mid ate$
$< NP > \quad -> < Det >< N >< PP >$	$< P > \quad -> in \mid on \mid to \mid into$
$< VP > \quad -> < V >$	$< Aux > \quad -> is \mid will \mid do$
$< VP > \quad -> < V >< NP >$	
$< VP > \quad -> < V >< NP >< PP >$	
$< PP > \quad -> < P >< NP >$	

- ✓ The symbol $->$ is used for 'defined as'.
- ✓ Vertical bar | for alternative definitions (OR)
- ✓ **S** for Sentence
- ✓ **N** for Noun
- ✓ **V** for Verb
- ✓ **Adj** for Adjective
- ✓ **NP** for Noun Phrase
- ✓ **VP** for Verb Phrase
- ✓ **P** for Preposition
- ✓ **PP** for Preposition Phrase

Hits:

- ✓ Each Noun belongs to Noun Phrase.
- ✓ Verb cannot be Noun Phrase.
- ✓ Before the Verb Phrase all are Noun Phrase.
- ✓ After the Verb Phrase all are Verb Phrase.

The simple sentence recognized by this grammar is:

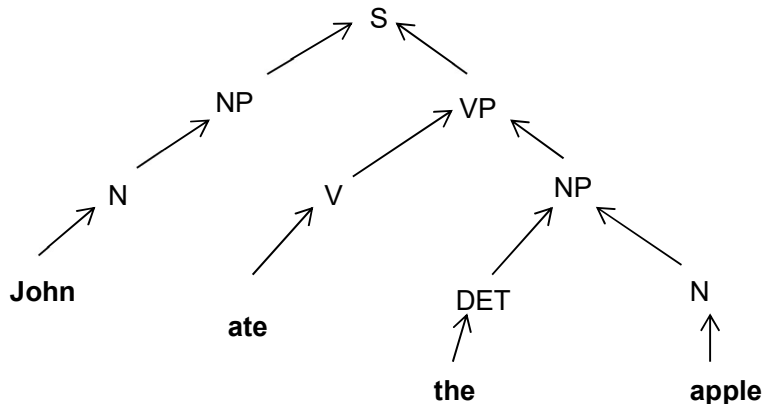
a) John ate the apple.

b) The cute girl ate an apple.

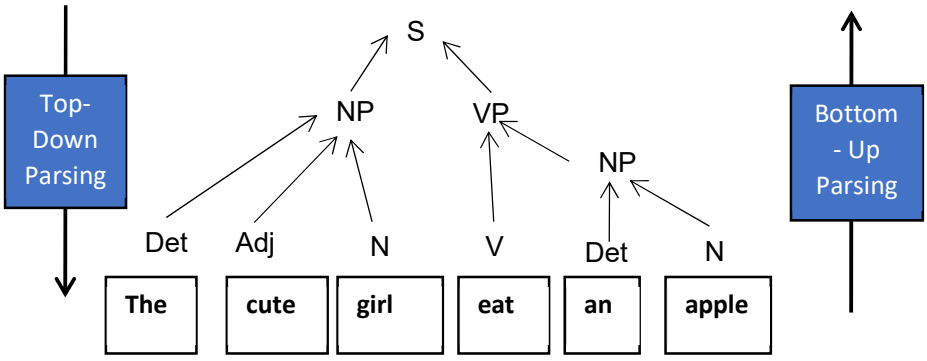
c) The cute girl sings a song.

A Parse Tree

i. John ate the apple.

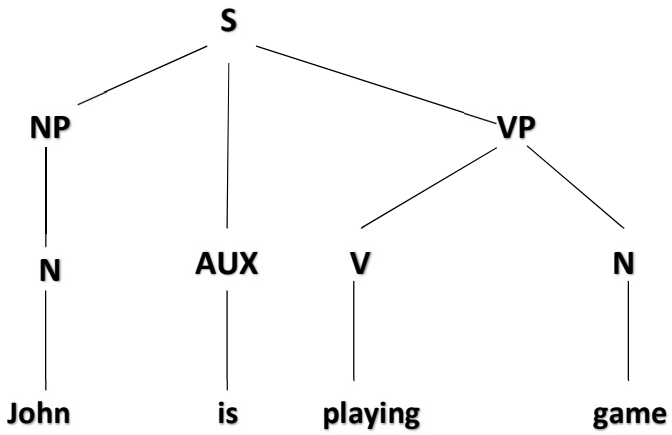


ii. The cute girl ate an apple

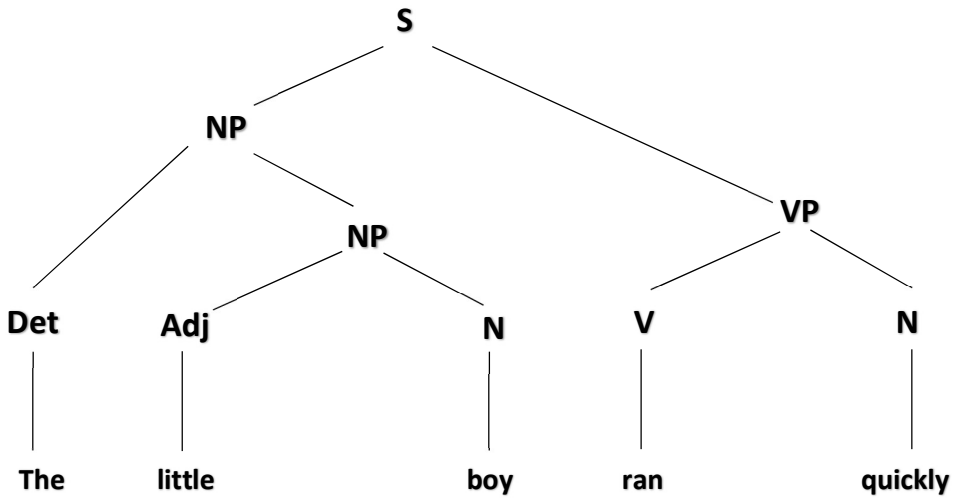


iii. The cute girl sings a song. *(Do it Yourself)*

iv. John is playing game.



v. The little boy ran quickly.



[Just change the arrow for bottom-up or top down parsing, if it is asked in exam]

6.3.1 Levels of Analysis: Phonetic, Syntactic, Semantic, Pragmatic

Natural Language Processing — Terminologies

- **Phonetics/Phonology:** It's a study of organizing sound. How words are pronounced in terms of sequences of sound.
- **Morphology:** Basically, it's a study of the construction of words from primitive meaningful units. Individual words are analyzed into their components and non-word tokens such as punctation are separated from words.
- **Morpheme:** As we can say that it's a primitive unit of meaning in a language.
- **Syntax:** In this, we have to arrange words to make a sentence. Also, involves determining the structural role of words. That is in the sentence and in phrases.
- **Semantics:** It defines the meaning of words. Moreover, how to combine words into meaningful phrases and sentences.
- **Pragmatics:** It deals with use and understanding sentences in different situations. Also, defines how the interpretation of the sentence is affected.
- **World Knowledge:** It includes general knowledge about the world.
- **Discourse:** It deals with how the immediately preceding sentences can affect the interpretation of the next sentence.

6.4 Introduction to Machine Vision

Artificial Intelligence (AI) is one of the most coveted technologies across the world. Since **John McCarthy** coined the term **'Artificial Intelligence'** for the first time, this technology has been unlocking possibilities for the mankind almost every single day. In the years to come, AI is sure to change the way we look at things; especially machines or devices.

In short, AI is enabling machines to analyze and think just like we humans do. Unlike before, now machines can think on their own, and respond accordingly. But this is not the end!

As a technology, AI is making advancements and its presence felt in almost all the verticals. Machine Vision is one among the newest trends AI that would surely make us rethink on the way we look at machines today.

What is Machine Vision?

Machine vision (MV) is an integrated mechanical-optical-electronic-software technology which makes use of optical instrumentation, digital video, electromagnetic sensing, mechanics and image processing technology. The technology's goal is optical and non-contact sensing to receive and analyze a real image in order to provide more information. Machine vision technology is widely used in monitoring and controlling a wide range of applications.

In technical terms, Machine Vision (MV) can be defined as a technology that enables a machine to see; to inspect, identify, and evaluate objects at rest or in motion. Though this may sound quite similar to the surveillance cameras, it should be mistaken as a similar one.

While surveillance cameras mostly work on image processing, Machine Visions enables cameras to detect the slightest of motions; may be by a micro millimeter, to recognize facial differences even at almost zero-light exposure, identify faces accurately, and evaluate still and motion images within a fraction of time.

But Why Do We Need Machines to See?

Human vision is limited to certain conditions and parameters. Usually, human vision is limited within a wavelength range of 380-740 nanometers. Anything below or above this range cannot be picked up by the human eye. But with machines, the range of visibility wavelength can be extended much further than the human range. In addition to increasing vision limit, this would also ensure better visibility, precision, and faster processing.

Now, combining this vision with Artificial Intelligence, one can match numerous patterns to identify objects in real-time. For example, packaging errors in packaging industry; even the minutest one that might not be visible with naked human eyes, can be identified by means of Machine Vision (MV). Thus, process efficiency can be enhanced with the correct implementation of MV in the production line.

Another great example would be integrating the surveillance cameras with Machine Vision to detect any unwanted person around one's premises. Though surveillance cameras can do this job, integrating them with MV would add a newer edge to it by enabling the cameras to detect trespassers, compare the images received with millions of faces in the data base, and send the information thus received to the property owner. With Machine Vision (MV), surveillance cameras can even detect the closest-matching figures even from silhouettes. Thus, a Machine Vision system can redefine and bolster the traditional security or surveillance methods.

Some Applications of Machine Vision (MV) Systems:

In Defense:

Drones are one of the most relied devices in defense sector. Equipping these drones with Machine Vision would provide information about the slightest of motion to the forces. Furthermore, the MV systems can also detect enemy movements, help to detect landmine fields, or any kind of ambush from the other side. The same can also be used to conduct real-time surveillance across locations that could not be accessed or require much effort.

In Robotics:

At present, almost all the rovers used for space research are laced with Machine Vision Systems; especially the ones used for Mars exploration. Unlike before, these modern-day rovers can find their own ways, detect obstacles on the way, recourse their way if required, and most importantly, they relay real-time images and videos to the space station. MV systems used in these rovers can even help the scientists detect water, keep a close eye on the temperature level, identify the available gases across those atmospheres, and lot more even while sitting millions of miles away from the rovers.

In Real Estate:

Security is one of the most commonly sought-after parameters among the real estate buyers. Developers these days, are using MV systems to ensure better and advanced security around their properties. In addition to detecting the visitors' faces, these MV Systems can predict the probable actions or intentions of the visitors by matching their course of action across millions of patterns, and notify the authorities in case there are any threats detected.

In Banking:

Face detection is already one of the most effective technology used in banking sector. You can read our blog on 'Face Scanning' here. Taking face scanning a step further is the Machine Vision technology. With this advanced AI-technology, banks can now not only scan the faces, but also match or compare the respective faces with millions of others and detect any of the slightest fraudulent activities if any.

6.5 Current trends and the future

ANI: Today's Artificial Intelligence

AI that is available right now: it's called **Artificial Narrow Intelligence** (ANI). ANI, also known as applied AI, is able to perform domain-specific reasoning tasks and problem solving, such as:

Speech and Text Recognition: Thanks to Natural Language Processing (NLP), a subfield of AI, program computers can process and analyze human (natural) languages, making possible the interactions between computers and humans in natural language. One of the most common applications of NLP is in voice assistants**: Amazon's Alexa, Apple's Siri, Microsoft's Cortana, etc.**

Computer Vision: The ability of a computer to process, analyze and understand digital images through the transformation of the visual input. The applications of computer vision are various: object or facial recognition, image restoration, motion analysis, etc.

Autonomous Vehicles: Autonomous vehicles, like these being developed by Tesla. Self-driving cars, although yet not so common on our roads, are an excellent example of the impact of AI on everyday life. Multiple AI subfields must be put together in order to make a vehicle that will replace human-driven cars: deep learning, speech recognition, voice search, image recognition and processing, motion detection, etc.

Product Recommendation Systems: Product recommendation systems, widely used by ecommerce shops. AI-powered recommendation engines analyze consumer behavior and preferences to deliver accurate, personalized product recommendations.

Ecommerce giants have successfully implemented AI recommendations in their marketing strategies: for instance, 35% percent of Amazon's revenue is generated by its recommendation's engine, and Zalando uses content-based and collaborative filtering algorithms to provide users with tailor-made product recommendations.

These problems require processing huge amount of data, something impossible for a human brain, but easily done for a well-trained AI algorithm. Nevertheless, it doesn't mean that a computer program performing these tasks is fully intelligent, as an algorithm built and trained for one purpose won't be able to perform well in a completely different task. This limitation

makes it hard to recognize current AI applications as a true AI. Actually, in 2017, intelligence tests conducted with some popular and publicly available ANI such as Google AI and Siri revealed that these AI achieved a score of about 47 IQ points, which is a level of a six-year-old child.

However, the ANI is universally acknowledged to be only one, least developed of the possible types of AI. What are the other types and how intelligent can they become?

AGI: The Future of Artificial Intelligence

The increasing computing power of modern computers and vast amount of data available to feed the algorithms allow assuming that we're on the way to reach the next levels of AI and machine learning development.

The next-level AI that is believed to appear in the future is known as Artificial General Intelligence (AGI) Right now, there are no AGI solutions available on the market, which means that at the moment ANI is the only type of AI that the humanity is able to build, therefore any other AI types are purely theoretical. On the other hand, lots of resources are spent every year on research, and there are many theories on how the advanced types of AI can be built, and although with no success yet, it gives hope that someday it will be achieved.

In short, Artificial General Intelligence (also known **as Deep AI or Strong AI**) is believed to be able to mimic human intelligence and behavior so well that it will become indistinguishable from the human mind in terms of learning and understanding any task, therefore successfully surpassing Turing test. AGI wouldn't be limited to performing specific tasks, like ANI is, and could be used for multiple purposes. The concept of human-like intelligent machines can feel disturbing, and indeed, it has caused concerns: in 2015, an open letter on artificial intelligence was signed by a number of scientists and researchers, among them Stephen Hawking and Elon Musk. The letter emphasized the importance of using the increasingly advanced AI systems for the societal benefit, and researching "how to reap its benefits while avoiding potential pitfalls", in other words: avoiding potential dangers for the humanity.

However, even that there is a lot of effort and resources put into AGI research, it's not expected to be built in the next couple of years. In the most optimistic scenario, there are decades more research required to develop a true AI.

XAI: eXplainable Artificial Intelligence

Another future AI trend that is being talked about is the XAI – eXplainable Artificial Intelligence. The term hints that this type of AI would be able to explain itself. Nowadays, there are many industries and use cases where machine learning methods give results that aren't easy to interpret. For instance, data scientist sometimes can struggle to understand what has led AI to make a specific decision, let alone explain it to another person, for example the customer. The concept of XAI, an AI that would explain its own decision-making process, was born from the need of being able to interpret the algorithms' way of reasoning, as it also could allow to improve it and avoid its mistakes.

The main obstacle in the development of XAI is that neural networks mostly are not so easy to explain, as the layers return weights matrices. There are already some ideas being implemented, like heat maps for the weight matrices that improve neural networks interpretability. For sure more research will be done in this area in the upcoming years.

The evolution of deep learning

Deep learning surely will be still a hot topic in the next decades and one of the most significant machine learning trends, as there is still a lot of work that can be done in this field. We see new methods introduced each year, that perform better than other methods or allow to use deep learning in brand new areas. You can stay up to date with the recent developments and breakthrough machine learning trends and methods thanks to events such as the annual NeurIPS conference.

What are some of the most interesting trends in deep learning?

Capsule neural networks, or CapsNet: it's a type of neural network that can improve hierarchical relationships models to better mimic human neurons.

FractalNet: it's an attempt to build ultra-deep neural network as an alternative to deep residual learning.

Deep reinforcement learning: this ML trend mixes deep learning (i.e. deep neural networks) with reinforcement learning (Q-learning, actor critic, etc.) in order to create powerful algorithms that might be able to solve complex problems.

THE END

PU Questions Answers

MYCIN:

MYCIN was based on backward chaining and could identify various bacteria that could cause acute infections. It could also recommend drugs based on the patient's weight.

Here, we will explain the working of an expert system by taking an example of MYCIN ES. Below are some steps to build an MYCIN:

- Firstly, ES should be fed with expert knowledge. In the case of MYCIN, human experts specialized in the medical field of bacterial infection, provide information about the causes, symptoms, and other knowledge in that domain.
- The KB of the MYCIN is updated successfully. In order to test it, the doctor provides a new problem to it. The problem is to identify the presence of the bacteria by inputting the details of a patient, including the symptoms, current condition, and medical history.
- The ES will need a questionnaire to be filled by the patient to know the general information about the patient, such as gender, age, etc.
- Now the system has collected all the information, so it will find the solution for the problem by applying if-then rules using the inference engine and using the facts stored within the KB.
- In the end, it will provide a response to the patient by using the user interface.

SOM (Self-Organizing Map) Explanation in Short:

- SOM is algorithm that projects high-dimensional data onto a two-dimensional map.
- The projection preserves the topology of the data so that similar data items will be mapped to nearby locations on the map.
- SOM still have many practical applications in **pattern recognition, speech analysis, industrial and medical diagnostics, data mining**.
- Large quantity of goods quality representative training data required.

[Fall 2015]

Q.6 b) What is neural network? Define Perceptron. How is winning node chosen in Kohonen network, illustrate with following dataset: **[8]**

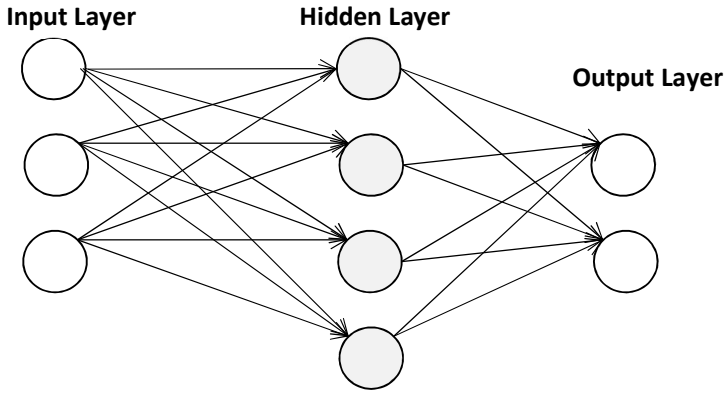
$$\boldsymbol{X} = \begin{bmatrix} \mathbf{0.52} \\ \mathbf{0.12} \end{bmatrix} \quad \boldsymbol{W_1} = \begin{bmatrix} \mathbf{0.52} \\ \mathbf{0.12} \end{bmatrix} \quad \boldsymbol{W_2} = \begin{bmatrix} \mathbf{0.27} \\ \mathbf{0.81} \end{bmatrix} \quad \boldsymbol{W_3} = \begin{bmatrix} \mathbf{0.43} \\ \mathbf{0.21} \end{bmatrix}$$

Where,

X two-dimensional input vector presented in 3-Dimensional Kohonen Network.

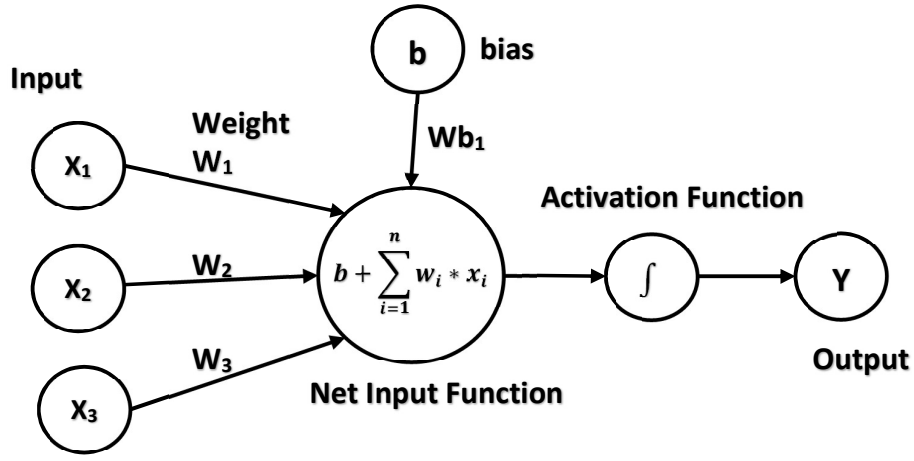
$$W_i = Weight\ Vectors, i = 1, 2, 3$$

Ans: A neural network is either a system software or hardware that works similar to the tasks performed by neurons of human brain. Neural networks include various technologies like deep learning, and machine learning as a part of Artificial Intelligence (AI). Artificial neural networks (ANN) is the key tool of machine learning. These are systems developed by the inspiration of neuron functionality in the brain, which will replicate the way we humans learn.



A perceptron is a neural network unit (an artificial neuron) that does certain computations to detect features or business intelligence in the input data. Perceptron was introduced by Frank Rosenblatt in 1957. He proposed a Perceptron learning rule based on the original MCP neuron.

A Perceptron is an algorithm for supervised learning of binary classifiers. This algorithm enables neurons to learn and processes elements in the training set one at a time.



Sol^{n}:

Given, the 2-Dimensional input vector X is represented to the neuron Kohonen network,

$$\boldsymbol{X} = \begin{bmatrix} \mathbf{0.52} \\ \mathbf{0.12} \end{bmatrix}$$

The intial weights, W_j are given by,

$$\boldsymbol{W_1} = \begin{bmatrix} \mathbf{0.52} \\ \mathbf{0.12} \end{bmatrix} \quad \boldsymbol{W_2} = \begin{bmatrix} \mathbf{0.27} \\ \mathbf{0.81} \end{bmatrix} \quad \boldsymbol{W_3} = \begin{bmatrix} \mathbf{0.43} \\ \mathbf{0.21} \end{bmatrix}$$

Now, find the wining (Best-Matching Neuron) j_x using the Minimum Euclidean Distance:

$$\boldsymbol{D(1)} = \sqrt{(\boldsymbol{X_1} - \boldsymbol{W_{11}})^2 + (\boldsymbol{X_2} - \boldsymbol{W_{21}})^2}$$

$$= \sqrt{(\mathbf{0.52} - \mathbf{0.27})^2 + (\mathbf{0.12} - \mathbf{0.81})^2} = \mathbf{0.73}$$

$$\boldsymbol{D(2)} = \sqrt{(\boldsymbol{X_1} - \boldsymbol{W_{12}})^2 + (\boldsymbol{X_2} - \boldsymbol{W_{22}})^2}$$

$$= \sqrt{(\mathbf{0.52} - \mathbf{0.42})^2 + (\mathbf{0.12} - \mathbf{0.70})^2} = \mathbf{0.59}$$

$$\boldsymbol{D(3)} = \sqrt{(\boldsymbol{X_1} - \boldsymbol{W_{13}})^2 + (\boldsymbol{X_2} - \boldsymbol{W_{23}})^2}$$

$$= \sqrt{(\mathbf{0.52} - \mathbf{0.43})^2 + (\mathbf{0.12} - \mathbf{0.21})^2} = \mathbf{0.13}$$ **(Minimum Euclidean Distance)**

$\therefore$ *Neuron* 3 *is the winner and its weight vector* W_3 *is updated according to the competitive learning rule.*

Assume, Learning rate $\alpha = 0.1$

The weights on the winning unit/neuron are updated:

$$\boldsymbol{W_{ij}(new)} = \boldsymbol{W_{ij}(old)} + \boldsymbol{\alpha}[\boldsymbol{X_i} - \boldsymbol{W_{ij}(old)}]$$

$$\begin{bmatrix} \boldsymbol{X_1} \\ \boldsymbol{X_2} \end{bmatrix} = \begin{bmatrix} \mathbf{0.52} \\ \mathbf{0.12} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{X_{13}} \\ \boldsymbol{X_{23}} \end{bmatrix} = \begin{bmatrix} \mathbf{0.43} \\ \mathbf{0.21} \end{bmatrix}$$

$$\boldsymbol{W_{13}(new)} = \boldsymbol{W_{13}(old)} + \mathbf{0.1}[\boldsymbol{X_1} - \boldsymbol{W_{13}(old)}]$$
$$\boldsymbol{W_{13}(new)} = \mathbf{0.43} + \mathbf{0.1}[\mathbf{0.52} - \mathbf{0.43}] = \mathbf{0.44}$$

$$\boldsymbol{W_{23}(new)} = \boldsymbol{W_{23}(old)} + \mathbf{0.1}[\boldsymbol{X_2} - \boldsymbol{W_{23}(old)}]$$
$$\boldsymbol{W_{23}(new)} = \mathbf{0.21} + \mathbf{0.1}[\mathbf{0.12} - \mathbf{0.21}] = \mathbf{0.20}$$

The weight vector W_3, of the winning neuron 3 becomes closer to the input vector X with each iteration/epoch.

[Fall 2016]

Q.6.a) Why is Natural Language understanding is difficult? Explain about Morphological Analysis with relevant example.

Ans: The primary problem with natural language processing is the ambiguity of language. There are a number of levels at which ambiguity may occur in natural language (of course a single sentence may include several of these levels). First a sentence or phrase may be ambiguous at a syntactic level. Syntax relates to the structure of the language, the way the words are put together. Some word sequences make valid sentences in a given language, some do not.

However, some sentence structures have more than one correct interpretation. In the first place, these are syntactically ambiguous. Secondly, a sentence may be ambiguous at a lexical level. The lexical level is the word and ambiguity here occur when a word can have more than one meaning. Thirdly, a sentence may be ambiguous at a referential level.

This is concerned with what the sentence (or a part of it) refers to. Ambiguity occurs when it is not clear what the sentence is referring to or it may legally refer to more than one thing. Fourthly, a sentence can be ambiguous at a semantic level, that is, at the point of the meaning of the sentence.

Morphological Analysis:

The study of word formulation – how words are built up from smaller pieces. Identification, analysis, and description of the structure of a given language's MORPHEMES and other linguistic units, such as root words, affixes, part of speech, intonations and stresses or implied context. Examples:

- Washing = Wash + ing
- Browser = Browse + er
- Rats = Rat + s

Individual words are analyzed into their components and non-word tokens such as punctuation are separated from the words.

Tries to extract root word from decline or inflectional form of word after removing suffices and prefixes. Ex: getting the root "push" from declined from pushes, pushed, pushing, etc.

Assign appropriate syntactic categories such as noun, verb, adjective etc. to all words in the sentence.

Types of Morphology:

a. **Inflectional Morphology:** Modification of a word to express different grammatical categories. Examples: Cats, Men etc.
b. **Derivational Morphology:** Creation of a new word from existing word by changing grammatical category. Examples: Happiness, Brotherhood etc.

[Fall 2018]

Q.5.b) Solve the AND function using Hebb network. **[8]**

Ans: Hebb Network:

- This rule, one of the oldest and simplest, was introduced by **Donald Hebb** in his book The Organization of Behavior in **1949**. It is a kind of feed-forward, unsupervised learning.
- Basic Concept − This rule is based on a proposal given by Hebb, who wrote − "When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased."
- From the above postulate, we can conclude that the connections between two neurons might be strengthened if the neurons fire at the same time and might weaken if they fire at different times.
- Mathematical Formulation − According to Hebbian learning rule, following is the formula to increase the weight of connection at every time step.

$$\Delta W_{ji}(t) = \alpha X_i(t).Y_j(t)$$

Here,

$\Delta W_{ji}(t) =$ increment by which the weight of connection increases at time step **t**

$\alpha =$ the positive and constant learning rate

$X_i(t) =$ the input value from pre-synaptic neuron at time step **t**

$Y_j(t) =$ the output of pre-synaptic neuron at same time step **t**

In short:

- Hebb learning rule is the simplest one.
- Hebb rule is suited for bipolar data (+1, -1).
- The learning in the brain is performed by the change in the synaptic gap.
- When an axon of cell A is near enough to excite cell B and repeatedly keep firing it, some growth process takes place in one or both cells.
- According to Hebb rule, weight vector is found to increase proportionately to the product of the input and learning signal.

$$W_i(new) = W_i(old) + X_iY$$

- Here learning signal is equals to neuron output.

Hebb Network Algorithm:

Step 0: Initialize all weight and bias to zero.

$$W_i = 0, \quad b = 0, \quad for\ i = 1\ to\ n$$

Step 1: For each of the training sample **s:t** *[i.e. pair (s, t)]***,** do step 2 to 4

s is the input pattern, **t** is the target output of the sample

Step 2: Set **s** to input units

$$X_i = S_i \qquad for\ i = 1\ to\ n$$

Step 3: Set **Y** to the target **t**

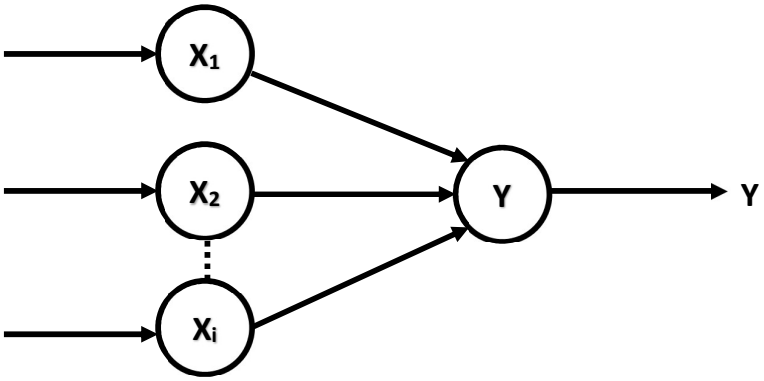
$$Y = t$$

Step 4: Update the weight and bias by applying Hebb Rule:

$$W_i(new) = W_i(old) + X_i * Y \qquad for\ i = 1\ to\ n$$

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

The algorithm requires only one pass through this training set



Solve the **AND function** using Hebb network.

Hebb rule is suited for bipolar unit (+1, -1). So, we do +1 for +1 and 0 for -1

[METHOD – 1] First draw the table like below:

Input Unit			Output	New W_1	New W_2	New b
X_1	X_2	b=1	Y=t			
1	1	1	1			
1	-1	1	-1			
-1	1	1	-1			
-1	-1	1	-1			

Initially, we have **W_i=0 and b=0**

Now, calculate $\boldsymbol{W_i(new)\ for\ i = 1\ to\ n}$

$\boldsymbol{W_i(new) = W_i(old) + X_i * Y}$

$\boldsymbol{W_1(new) = W_1(old) + X_1 * Y}$

Initially $W_i = 0$

$W_1(old)$	+	X_1	*	Y	=	$W_1(new)$	$W_1 = 0$
0	+	1	*	1	=	**1**	$W_1 = 1$
1	+	1	*	-1	=	**0**	$W_1 = 0$
0	+	-1	*	-1	=	**1**	$W_1 = 1$
1	+	-1	*	-1	=	**2**	

Value of W_1 *is updated from* 0 *to* 1

Value of W_1 *is updated from* 1 *to* 0

Value of W_1 *is updated from* 0 *to* 1

Now, calculate $\boldsymbol{W_i(new)\ for\ i = 1\ to\ n}$

$\boldsymbol{W_i(new) = W_i(old) + X_i * Y}$

$\boldsymbol{W_2(new) = W_2(old) + X_2 * Y}$

Initially $W_i = 0$

$W_2(old)$	+	X_2	*	Y	=	$W_2(new)$	$W_2 = 0$
0	+	1	*	1	=	**1**	$W_2 = 1$
1	+	-1	*	-1	=	**2**	$W_2 = 2$
2	+	1	*	-1	=	**1**	$W_2 = 1$
1	+	-1	*	-1	=	**2**	

Value of W_2 *is updated from* 0 *to* 1

Value of W_2 *is updated from* 1 *to* 2

Value of W_2 *is updated from* 2 *to* 1

Now, calculate $b(new)$

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

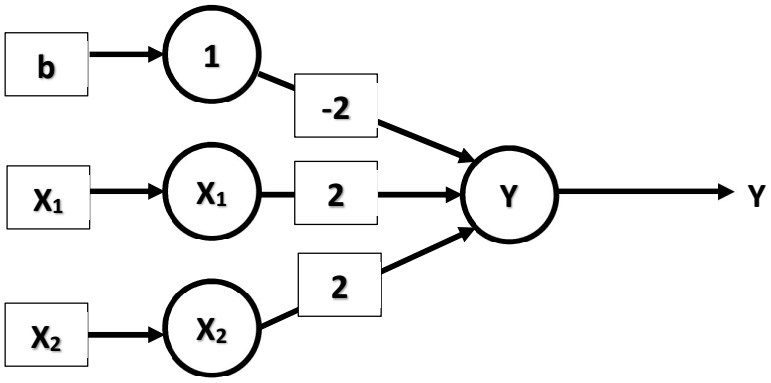
Initially $b = 0$

$b(old)$	+	Y	=	$b(new)$	$b = 0$	
0	+	1	=	1	$b = 1$	Value of b is updated from 0 to 1
1	+	-1	=	0	$b = 0$	Value of b is updated from 1 to 0
0	+	-1	=	-1	$b = -1$	Value of b is updated from 0 to -1
-1	+	-1	=	-2		

Now adjust the newly updated weight and bias into the table

Input Unit			Output	New W_1	New W_2	New b
X_1	X_2	b=1	Y=t			
1	1	1	1	1	1	1
1	-1	1	-1	0	2	0
-1	1	1	-1	1	1	-1
-1	-1	1	-1	2	2	-2

Hebb Network for AND Function



[METHOD – 2] First draw the table like below:

Input Unit			Output	ΔW_1	ΔW_2	Δb	New W_1	New W_2	New b
X_1	X_2	b=1	Y=t						
1	1	1	1						
1	-1	1	-1						
-1	1	1	-1						
-1	-1	1	-1						

Initially, the weights are set to zero and bias is also set as zero.

$W_1 = W_2 = b = 0$

Calculate,

$\Delta W_i = X_i Y$ and

$\Delta b = Y$ and fill the table

Input Unit			Output	ΔW_1	ΔW_2	Δb	New W_1	New W_2	New b
X_1	X_2	b=1	Y=t						
1	1	1	1	1	1	1			
1	-1	1	-1	-1	1	-1			
-1	1	1	-1	1	-1	-1			
-1	-1	1	-1	1	1	-1			

For ΔW_1

$\Delta W_1 = X_1 * Y$

1	*	1	=	1
1	*	-1	=	-1
-1	*	-1	=	1
-1	*	-1	=	1

For ΔW_2

$\Delta W_2 = X_2 * Y$

1	*	1	=	1
-1	*	-1	=	1
1	*	-1	=	-1
-1	*	-1	=	1

For Δb

$\Delta b = Y$

1	=	1
-1	=	-1
-1	=	-1
-1	=	-1

Now, calculate new weights and new bias

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

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

Input Unit			Output	ΔW_1	ΔW_2	Δb	New W_1	New W_2	New b
X_1	X_2	b=1	Y=t						
1	1	1	1	1	1	1			
1	-1	1	-1	-1	1	-1			
-1	1	1	-1	1	-1	-1			
-1	-1	1	-1	1	1	-1			

For $W_1(new)$

$W_1(new) = W_1(old) + \Delta W_1$

0	+	1	=	1
1	+	-1	=	0
0	+	1	=	1
1	+	1	=	2

Initial $W_1 = 0$

For $W_2(new)$

$W_2(new) = W_2(old) + \Delta W_2$

0	+	1	=	1
1	+	1	=	2
2	+	-1	=	1
1	+	1	=	2

Initial $W_2 = 0$

For $b(new)$

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

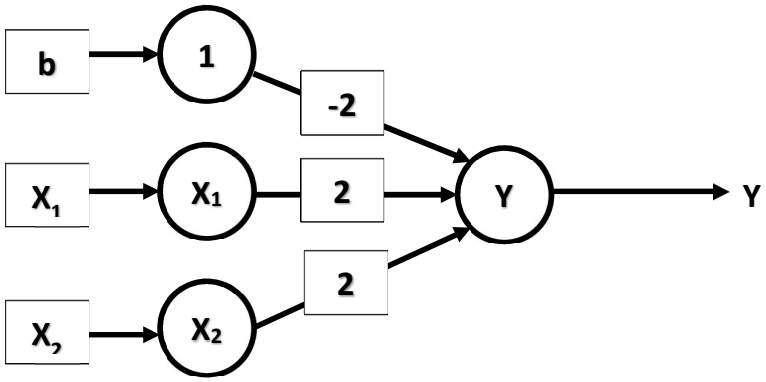
0	+	1	=	1
1	+	-1	=	0
0	+	-1	=	-1
1	+	-1	=	2

Initial b = 0

Final Table is:

Input Unit			Output	ΔW_1	ΔW_2	Δb	New W_1	New W_1	New b
X_1	X_2	b=1	Y=t						
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

Hebb Network for AND Function



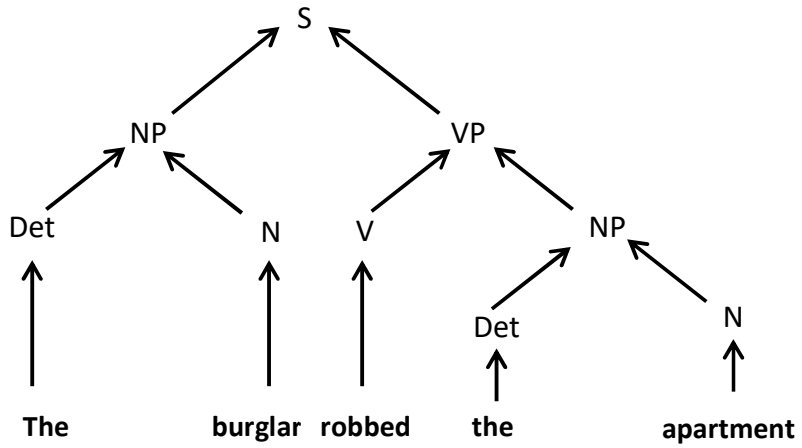
[Fall 2019]

Q.6.b) What do you mean by NLP? Construct a parse tree for the sentence **"The burglar robbed the apartment"**.

Ans: About NLP ***[Already discussed see above]***

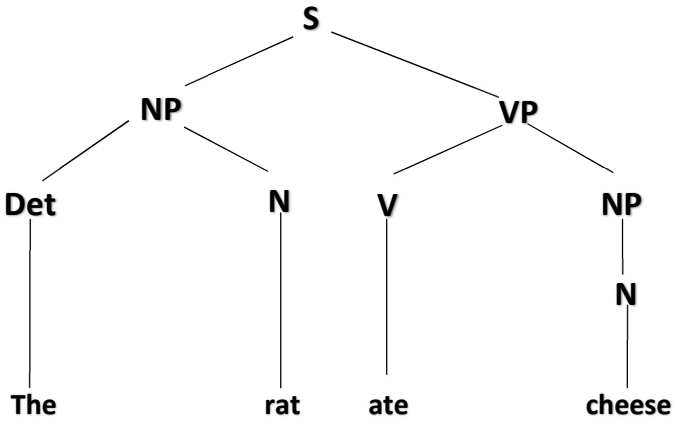
Parse Tree for **"The burglar robbed the apartment".**

Rules	Dictionary Words
$< S > \ -> \ < NP >< VP >$	$< Det > \ -> \ a \mid an \mid the \mid this$
$< NP > \ -> \ < Adj >< N >$	$< N > \ -> \ girl \mid apple \mid john$
$< NP > \ -> \ < Det >< N >$	$< Adj > \ -> \ cute \mid smart$
$< NP > \ -> \ < Det >< Adj >< N >$	$< V > \ -> \ sings \mid eats \mid ate$
$< NP > \ -> \ < Det >< N >< PP >$	$< P > \ -> \ in \mid on \mid to \mid into$
$< VP > \ -> \ < V >$	$< Aux > \ -> \ is \mid will \mid do$
$< VP > \ -> < V >< NP >$	
$< VP > \ -> < V >< NP >< PP >$	
$< PP > \ -> < P >< NP >$	

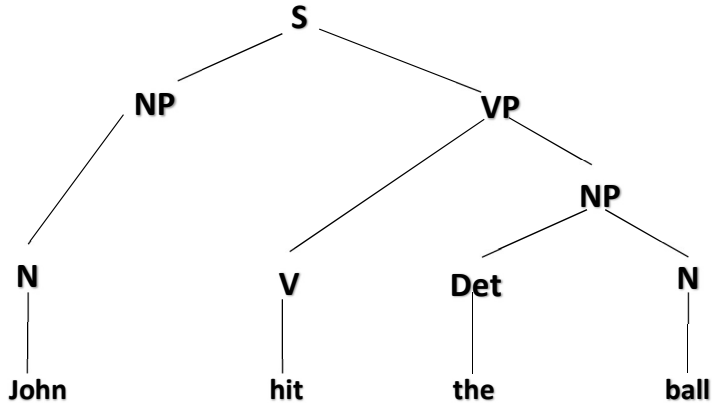


Some Practice on Parse Tree:

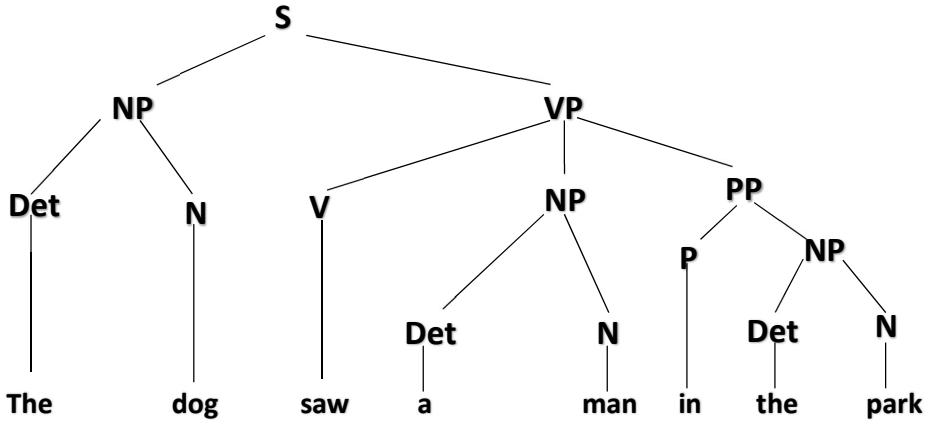
i. The rat ate cheese.



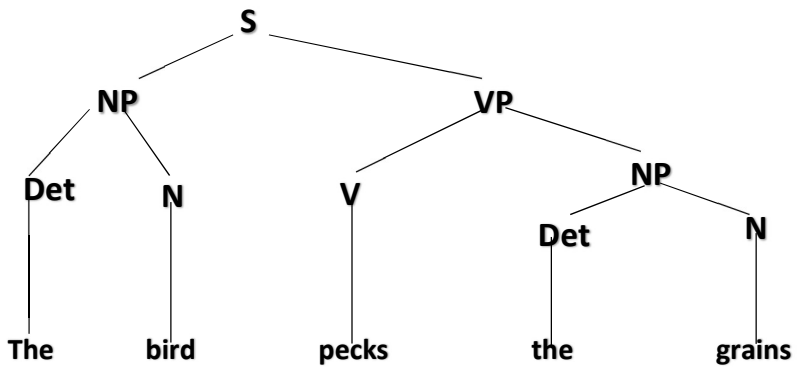
ii. John hit the ball.



iii. The dog saw a man in the park.



iv. The bird pecks the grains.



 # What is an Activation Function? Why to use them?

Ans: Neural network activation functions are a crucial component of deep learning. Activation functions determine the output of a deep learning model, its accuracy, and also the computational efficiency of training a model—which can make or break a large scale neural network.

Activation functions are mathematical equations that determine the output of a neural network. The function is attached to each neuron in the network, and determines whether it should be activated ("fired") or not, based on whether each neuron's input is relevant for the model's prediction. Activation functions also help normalize the output of each neuron to a range between 1 and 0 or between -1 and 1.

Activation function decides, whether a neuron should be activated or not by calculating weighted sum and further adding bias with it. The purpose of the activation function is to introduce non-linearity into the output of a neuron.

We know neural network has neurons that work in correspondence of weight, bias and their respective activation function. In a neural network, we would update the weight and bias of the neurons on the basis of the error at the output. This process is known as back-propagation. Activation functions make the back-propagation possible since the gradients are supplied along with the error to update the weights and biases

---

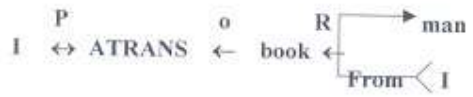
[Spring 2014]

Q.7) Write Short Notes on:

b) Conceptual dependencies:

- A Model of Natural Language understanding in Artificial Intelligence -Represent knowledge acquire from NL input.
- Conceptual Dependency (CD) theory was developed by Schank in 1973 to 1975 to represent the meaning of NL sentences.
- It helps in drawing inferences − It is independent of the language.
- CD representation of a sentence is not built using words in the sentence rather built using conceptual primitives which give the intended meanings of words.
- CD provides structures and specific set of primitives from which representation can be built.

I gave the man a book.
In CD from the above sentence can be represented with primitives as



It should be noted that this representation is same for different saying with same meaning.

For example − **I gave the man a book**, − **The man got book from me**, − **The book was given to man by me** etc.

In the above representation the symbols have the following meaning:

- ✓ Arrows indicate direction of dependency
- ✓ Double arrow indicates two may link between actor and the action
  - o P indicates past tense
  - o ATRANS is one of the primitive acts used by the theory. it indicates transfer of possession
  - o O indicates the object case relation
  - o R indicates the recipient case relation
  - o D indicates destination

Conceptual dependency provides a str5ucture in which knowledge can be represented and also a set of building blocks from which representations can be built. A typical set of primitive actions are

ATRANS - Transfer of an abstract relationship (Eg: give)

PTRANS - Transfer of the physical location of an object (Eg: go)

PROPEL - Application of physical force to an object (Eg: push)

MOVE - Movement of a body part by its owner (eg : kick)

GRASP - Grasping of an object by an actor (Eg: throw)

INGEST - Ingesting of an object by an animal (Eg: eat)

EXPEL - Expulsion of something from the body of an animal (cry)

MTRANS - Transfer of mental information (Eg: tell)

MBUILD - Building new information out of old (Eg: decide)

SPEAK - Production of sounds (Eg: say)

ATTEND - Focusing of sense organ toward a stimulus (Eg: listen)

A second set of building block is the set of allowable dependencies among the conceptualization describe in a sentence.

 Artificial Intelligence	 Machine Learning	 Deep Learning
Artificial intelligence originated around 1950s.	Machine learning originated around 1960s.	Deep learning originated around 1970s.
AI represents simulated intelligence in machines.	Machine Learning is the practice of getting machines to make decisions without being programmed.	Deep Learning is the process of using Artificial Neural Networks to solve complex problems.
AI is a subset of Data Science.	Machine learning is a subset of AI & Data Science	Deep learning is a subset of Machine learning, AI & Data Science.
Aim is to build machines which are capable of thinking like humans.	Aim is to make machines learn through data so that they can solve problems.	Aim is to build neural networks that automatically discover patterns for feature detection.

THE END
