MODULE-IV

SYLLABUS

Learning and Neural Networks: Learning from Observations, Inductive Learning and Learning Decision Trees. Learning in Neural and Belief Networks': neural networks, perceptrons, multilayer feed-forward networks and Bayesian methods for learning belief networks.

Learning from observations:

Inductive Learning

Inductive Learning, also known as Concept Learning, is how A.I. systems attempt to use a generalized rule to carry out observations.

Inductive Learning Algorithms (APIs) are used to generate a set of classification rules. These generated rules are in the "If this, then that" format.

These rules determine the state of an entity at each iteration step in Learning and how the Learning can be effectively changed by adding more rules to the existing ruleset.

When the output and examples of the function are fed into the A.I. system, inductive Learning attempts to learn the function for new data.

The Fundamental Concept of Inductive Learning

There are two methods for obtaining knowledge in the real world: first, from domain experts, and second, from machine learning.

Domain experts are not very useful or reliable for large amounts of data. As a result, we are adopting a machine learning approach for this project.

The other method, machine learning, replicates the logic of 'experts' in algorithms, but this work may be very complex, time-consuming, and expensive.

As a result, an option is the inductive algorithms, which generate a strategy for performing a task without requiring instruction at each step.

According to Jason Brownlee in his article "Basic Concepts in Machine Learning," an excellent method to understand how Inductive Learning works is, for example, if we are given input samples (x) and output samples (f(x)) from the perspective of inductive Learning, and the problem is to estimate the function (f).

It is necessary then to generalize from the samples and the mapping so that it can be used to estimate the output for new samples in the future.

In practice, estimating the function is almost always too complicated, so we seek excellent approximations.

Some practical examples of induction are:

Credit risk assessment.

- The x is the property of the customer.
- The f(x) is credit approved or not.

Disease diagnosis.

- The x is the characteristics of a given patient.
- The f(x) is the patient's disease.

Face recognition.

- The x are bitmaps of the faces we want to recognize.
- The f(x) is a name assigned to that face.

Automatic steering (autonomous driving).

- The x is bitmap images from a camera in front of the car.
- The f(x) is the degree to which the steering wheel should be turned.

Application

There are some situations in which inductive Learning is not a good idea. Therefore, understanding when and when not to use supervised machine learning is critical.

Inductive Learning may be helpful in the following four situations:

- **Problems in which no human expertise is available**. People cannot write a program to solve a problem if they do not know the answer. These are areas ripe for exploration.
- **Humans can complete the task, but no one knows how to do it.** There are situations in which humans can do things that computers cannot or do not do well. Riding a bike or driving a car are two examples.
- **Problems where the desired function is frequently changing**. Humans could describe it and write a program to solve it, but the problem changes too frequently. It is not economical. The stock market is one example.
- **Problems where each user requires a unique function.** Writing a custom program for each user is not cost-effective. Consider Netflix or Amazon recommendations for movies or books.

During the last years, there has been an increase in the amount of research on inductive Learning and its applications to various domains.

Concept learning is the most common application of inductive Learning.

Concept learning aims to find symbolic descriptions expressed in high-level terms that people can understand.

The following is a definition of concept learning:

- In addition, a set of (positive and negative) examples of a concept and possibly some background knowledge is provided.
- A general description (hypothesis) of the concept describes all of the positive examples but none of the negative ones.

MARBLE, a knowledge-based artificial intelligence (A.I.) system, has been developed to assess the riskiness of business loan applicants.

The paper "**Applying inductive learning to enhance knowledge-based expert systems**" describes the use of inductive Learning in MARBLE, a knowledge-based expert system developed to aid business loan evaluation. MARBLE is unique in that it can learn.

Thus, the A.I. system employs inference rules to simulate a lending officer's thought process when evaluating a loan request, generating loan-granting decision rules based on historical and proforma financial data.

This paper presents a learning method for inducing decision rules from training examples.

In the paper "**Inductive learning for risk classification**," the authors discuss the application of inductive Learning to credit risk analysis, a similar domain application.

The authors address three risk classification issues:

- Business loan evaluation
- Bond-rating
- Prediction of Bankruptcy

It is discussed how to use the previously mentioned Marble system, a knowledge-based decision-support system that employs approximately 80 decision rules to evaluate commercial loans.

The paper describes an aspect of Marble that uses inductive Learning to classify financial risks and discusses the technique's effectiveness.

Another example we can find in the paper "**On the Application of Inductive Machine Learning Tools to Geographical Analysis**" discusses the role of inductive machine learning in geographical analysis.

The presented discussion is not based on comparative results or mathematical descriptions but rather is based on how the various inductive learning approaches differ operationally, describing:

- How the feature space is partitioned or clustered.
- To find suitable solutions, search mechanisms are used.
- The multiple biases imposed by each technique.

When considering complex geographic feature spaces, the consequences of these issues are then detailed.

The goal is to provide the basis for developing reliable inductive analysis methods rather than relying on piecemeal or haphazard experimentation with the various operational criteria that inductive learning tools require.

Inductive Learning has also been used in education. For example, because global education allows students to obtain education from multiple education providers through various study exchange programs, it necessitates comparing available study courses in foreign institutions to courses on the institution's curriculum that issue the degree.

This issue necessitates a manual course comparison, which can be time-consuming and necessitates highly skilled experts to avoid the comparison, resulting in a superficial intuitive judgment and, as a result, course incompatibility issues.

The study, "**Interactive Inductive Learning: Application in Domain of Education.**" presents a technique for using Interactive inductive Learning supported by enterprise modeling to support mechanisms that can help save time and effort in study course comparison.

Because inductive learning algorithms are domain-independent, they can be used in any classification or pattern recognition task.

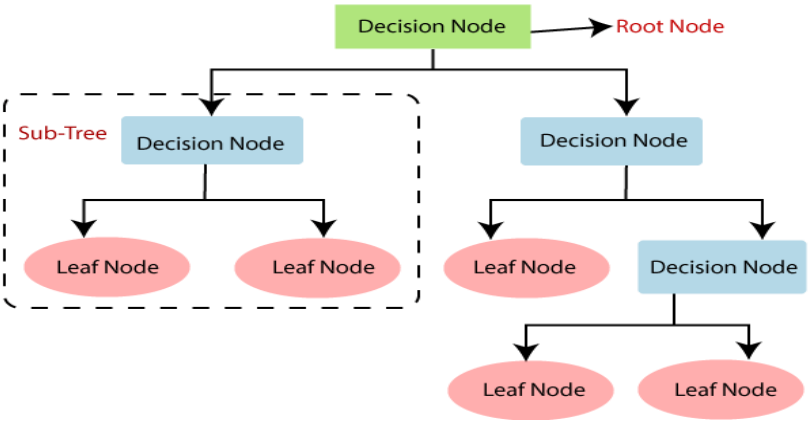
Several applications of the rules family of induction algorithms to visual inspection are presented in the paper entitled "**Applications of Inductive Learning to Automated Visual Inspection.**"

The primary value of using Inductive Learning for visual inspection, according to authors Mehmet Sabih Aksoy, Orhan Torkul, Abdullah Almudimigh, and Ismail H. Cedimoglu:

- The systems have no orientation issues, which are critical in digital image processing.
- Because rules represent the pattern, it is not necessary to store it in graphics form in memory. Instead, memory space is saved as a result of this.
- Because the number of conditions in each rule and the total number of rules is insignificant, the decision can be reached quickly.
- Because these systems are not very complex, developing software and designing hardware for them is simpler.

Decision Tree

- Decision Tree is a **Supervised learning technique** that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome.**
- In a Decision tree, there are two nodes, which are the **Decision Node** and **Leaf Node.** Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset.
- ***It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.***
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- In order to build a tree, we use the **CART algorithm,** which stands for **Classification and Regression Tree algorithm.**
- A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.
- Below diagram explains the general structure of a decision tree:



Why use Decision Trees?

There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree:

- Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
- The logic behind the decision tree can be easily understood because it shows a tree-like structure.

Decision Tree Terminologies

 Root Node: Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.

 Leaf Node: Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.

 Splitting: Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.

 Branch/Sub Tree: A tree formed by splitting the tree.

 Pruning: Pruning is the process of removing the unwanted branches from the tree.

 Parent/Child node: The root node of the tree is called the parent node, and other nodes are called the child nodes.

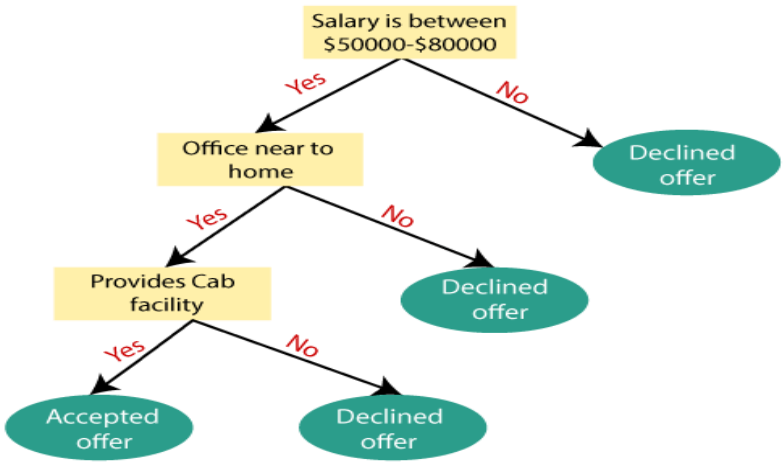
How does the Decision Tree algorithm Work?

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

- **Step-1:** Begin the tree with the root node, says S, which contains the complete dataset.
- **Step-2:** Find the best attribute in the dataset using **Attribute Selection Measure (ASM).**
- **Step-3:** Divide the S into subsets that contains possible values for the best attributes.
- **Step-4:** Generate the decision tree node, which contains the best attribute.
- **Step-5:** Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

Example: Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM). The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels. The next decision node further gets split into one decision node (Cab facility) and one leaf node. Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer). Consider the below diagram:



Attribute Selection Measures

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as **Attribute selection measure or ASM.** By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

- **Information Gain**
- **Gini Index**

1. Information Gain:

- Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
- It calculates how much information a feature provides us about a class.
- According to the value of information gain, we split the node and build the decision tree.
- A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)

Entropy: Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)

Where,

- **S= Total number of samples**
- **P(yes)= probability of yes**
- **P(no)= probability of no**

2. Gini Index:

- Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm.
- An attribute with the low Gini index should be preferred as compared to the high Gini index.
- It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.
- Gini index can be calculated using the below formula:

Gini Index= $1- \sum_j P_j^2$

Pruning: Getting an Optimal Decision tree

Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree.

A too-large tree increases the risk of overfitting, and a small tree may not capture all the important features of the dataset. Therefore, a technique that decreases the size of the learning tree without reducing accuracy is known as Pruning. There are mainly two types of tree **pruning** technology used:

- **Cost Complexity Pruning**
- **Reduced Error Pruning.**

Advantages of the Decision Tree

- It is simple to understand as it follows the same process which a human follow while making any decision in real-life.
- It can be very useful for solving decision-related problems.
- It helps to think about all the possible outcomes for a problem.
- There is less requirement of data cleaning compared to other algorithms.

Disadvantages of the Decision Tree

- The decision tree contains lots of layers, which makes it complex.
- It may have an overfitting issue, which can be resolved using the **Random Forest algorithm.**
- For more class labels, the computational complexity of the decision tree may increase.

Artificial Neural Network

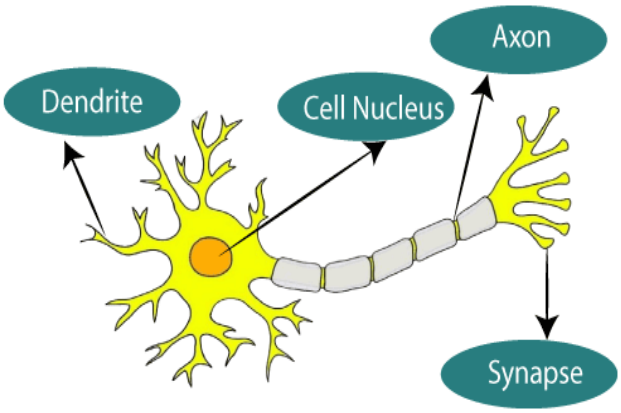
Artificial Neural Network Tutorial provides basic and advanced concepts of ANNs. Our Artificial Neural Network tutorial is developed for beginners as well as professions.

The term "Artificial neural network" refers to a biologically inspired sub-field of artificial intelligence modeled after the brain. An Artificial neural network is usually a computational network based on biological neural networks that construct the structure of the human brain. Similar to a human brain has neurons interconnected to each other, artificial neural networks also have neurons that are linked to each other in various layers of the networks. These neurons are known as nodes.

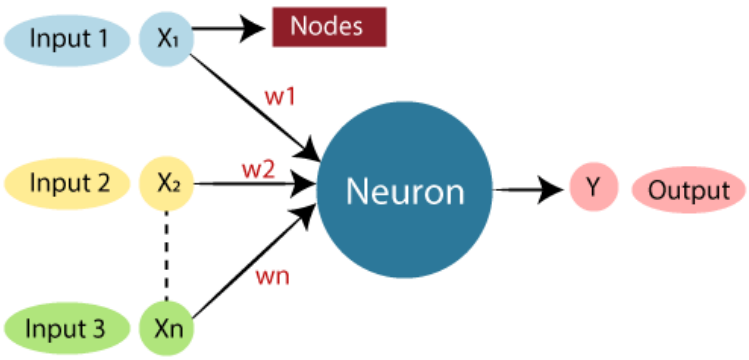
Artificial neural network tutorial covers all the aspects related to the artificial neural network. In this tutorial, we will discuss ANNs, Adaptive resonance theory, Kohonen self-organizing map, Building blocks, unsupervised learning, Genetic algorithm, etc.

What is Artificial Neural Network?

The term "**Artificial Neural Network**" is derived from Biological neural networks that develop the structure of a human brain. Similar to the human brain that has neurons interconnected to one another, artificial neural networks also have neurons that are interconnected to one another in various layers of the networks. These neurons are known as nodes.



The given figure illustrates the typical diagram of Biological Neural Network.
The typical Artificial Neural Network looks something like the given figure.



Dendrites from Biological Neural Network represent inputs in Artificial Neural Networks, cell nucleus represents Nodes, synapse represents Weights, and Axon represents Output.

Relationship between Biological neural network and artificial neural network:

Biological Neural Network	Artificial Neural Network
Dendrites	Inputs
Cell nucleus	Nodes
Synapse	Weights
Axon	Output

An **Artificial Neural Network** in the field of **Artificial intelligence** where it attempts to mimic the network of neurons makes up a human brain so that computers will have an option to understand things and make decisions in a human-like manner. The artificial neural network is designed by programming computers to behave simply like interconnected brain cells.

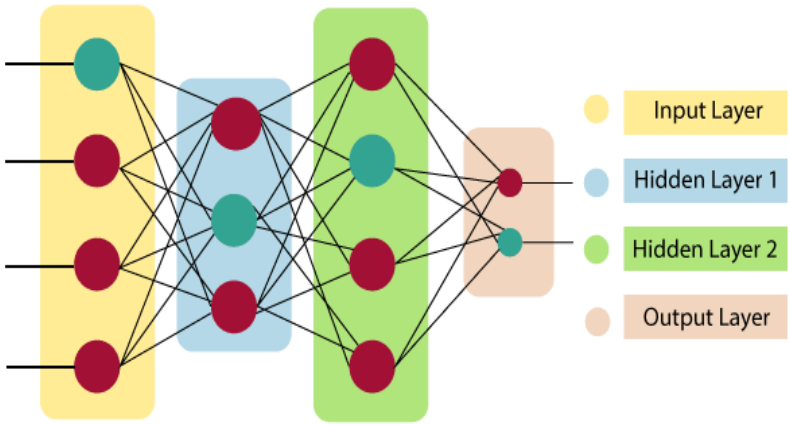
There are around 1000 billion neurons in the human brain. Each neuron has an association point somewhere in the range of 1,000 and 100,000. In the human brain, data is stored in such a manner as to be distributed, and we can extract more than one piece of this data when necessary from our memory parallelly. We can say that the human brain is made up of incredibly amazing parallel processors.

We can understand the artificial neural network with an example, consider an example of a digital logic gate that takes an input and gives an output. "OR" gate, which takes two inputs. If one or both the inputs are "On," then we get "On" in output. If both the inputs are "Off," then we get "Off" in output. Here the output depends upon input. Our brain does not perform the same task. The outputs to inputs relationship keep changing because of the neurons in our brain, which are "learning."

The architecture of an artificial neural network:

To understand the concept of the architecture of an artificial neural network, we have to understand what a neural network consists of. In order to define a neural network that consists of a large number of artificial neurons, which are termed units arranged in a sequence of layers. Lets us look at various types of layers available in an artificial neural network.

Artificial Neural Network primarily consists of three layers:

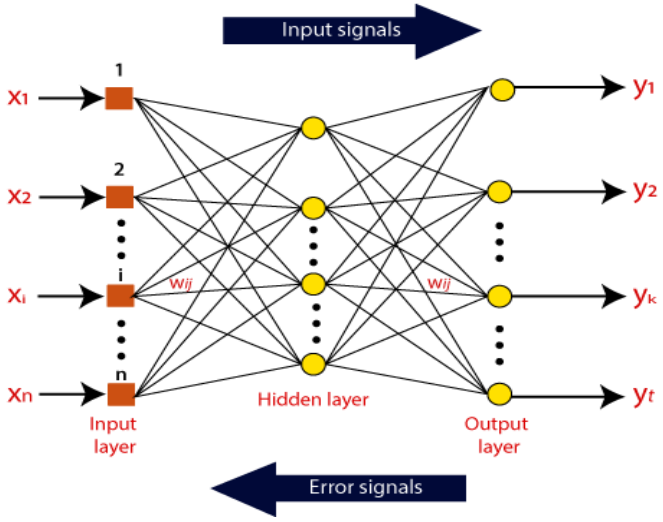


Input Layer:
As the name suggests, it accepts inputs in several different formats provided by the programmer.
Hidden Layer:
The hidden layer presents in-between input and output layers. It performs all the calculations to find hidden features and patterns.
Output Layer:
The input goes through a series of transformations using the hidden layer, which finally results in output that is conveyed using this layer.
The artificial neural network takes input and computes the weighted sum of the inputs and includes a bias. This computation is represented in the form of a transfer function.

$$\sum_{i=1}^{n} Wi * Xi + b$$

It determines weighted total is passed as an input to an activation function to produce the output. Activation functions choose whether a node should fire or not. Only those who are fired make it to the output layer. There are distinctive activation functions available that can be applied upon the sort of task we are performing.
Advantages of Artificial Neural Network (ANN)
Parallel processing capability:
Artificial neural networks have a numerical value that can perform more than one task simultaneously.
Storing data on the entire network:
Data that is used in traditional programming is stored on the whole network, not on a database. The disappearance of a couple of pieces of data in one place doesn't prevent the network from working.
Capability to work with incomplete knowledge:
After ANN training, the information may produce output even with inadequate data. The loss of performance here relies upon the significance of missing data.
Having a memory distribution:
For ANN is to be able to adapt, it is important to determine the examples and to encourage the network according to the desired output by demonstrating these examples to the network. The succession of the network is directly proportional to the chosen instances, and if the event can't appear to the network in all its aspects, it can produce false output.
Having fault tolerance:
Extortion of one or more cells of ANN does not prohibit it from generating output, and this feature makes the network fault-tolerance.
Disadvantages of Artificial Neural Network:
Assurance of proper network structure:
There is no particular guideline for determining the structure of artificial neural networks. The appropriate network structure is accomplished through experience, trial, and error.

Unrecognized behavior of the network:
It is the most significant issue of ANN. When ANN produces a testing solution, it does not provide insight concerning why and how. It decreases trust in the network.
Hardware dependence:
Artificial neural networks need processors with parallel processing power, as per their structure. Therefore, the realization of the equipment is dependent.
Difficulty of showing the issue to the network:
ANNs can work with numerical data. Problems must be converted into numerical values before being introduced to ANN. The presentation mechanism to be resolved here will directly impact the performance of the network. It relies on the user's abilities.
The duration of the network is unknown:
The network is reduced to a specific value of the error, and this value does not give us optimum results.
How do artificial neural networks work?
Artificial Neural Network can be best represented as a weighted directed graph, where the artificial neurons form the nodes. The association between the neurons outputs and neuron inputs can be viewed as the directed edges with weights. The Artificial Neural Network receives the input signal from the external source in the form of a pattern and image in the form of a vector. These inputs are then mathematically assigned by the notations x(n) for every n number of inputs.



Afterward, each of the input is multiplied by its corresponding weights (these weights are the details utilized by the artificial neural networks to solve a specific problem). In general terms, these weights normally represent the strength of the interconnection between neurons inside the artificial neural network. All the weighted inputs are summarized inside the computing unit.
If the weighted sum is equal to zero, then bias is added to make the output non-zero or something else to scale up to the system's response. Bias has the same input, and weight equals to 1. Here the total of weighted inputs can be in the range of 0 to positive infinity. Here, to keep the response in the limits of the desired value, a certain maximum value is benchmarked, and the total of weighted inputs is passed through the activation function.
The activation function refers to the set of transfer functions used to achieve the desired output. There is a different kind of the activation function, but primarily either linear or non-linear sets of functions. Some of the commonly used sets of activation functions are the Binary, linear, and Tan hyperbolic sigmoidal activation functions. Let us take a look at each of them in details:
Binary:
In binary activation function, the output is either a one or a 0. Here, to accomplish this, there is a threshold value set up. If the net weighted input of neurons is more than 1, then the final output of the activation function is returned as one or else the output is returned as 0.

Sigmoidal Hyperbolic:

The Sigmoidal Hyperbola function is generally seen as an "**S**" shaped curve. Here the tan hyperbolic function is used to approximate output from the actual net input. The function is defined as:

F(x) = (1/1 + exp(-????x))

Where ???? is considered the Steepness parameter.

Types of Artificial Neural Network:

There are various types of Artificial Neural Networks (ANN) depending upon the human brain neuron and network functions, an artificial neural network similarly performs tasks. The majority of the artificial neural networks will have some similarities with a more complex biological partner and are very effective at their expected tasks. For example, segmentation or classification.

Feedback ANN:

In this type of ANN, the output returns into the network to accomplish the best-evolved results internally. As per the **University of Massachusetts**, Lowell Centre for Atmospheric Research. The feedback networks feed information back into itself and are well suited to solve optimization issues. The Internal system error corrections utilize feedback ANNs.

Feed-Forward ANN:

A feed-forward network is a basic neural network comprising of an input layer, an output layer, and at least one layer of a neuron. Through assessment of its output by reviewing its input, the intensity of the network can be noticed based on group behavior of the associated neurons, and the output is decided. The primary advantage of this network is that it figures out how to evaluate and recognize input patterns.

Prerequisite

No specific expertise is needed as a prerequisite before starting this tutorial.

Audience

Our Artificial Neural Network Tutorial is developed for beginners as well as professionals, to help them understand the basic concept of ANNs.

Problems

We assure you that you will not find any problem in this Artificial Neural Network tutorial. But if there is any problem or mistake, please post the problem in the contact form so that we can further improve it.

Perceptron model

Perceptron is an algorithm for supervised learning of various binary classification tasks. Further, ***Perceptron is also understood as an Artificial Neuron or neural network unit that helps to detect certain input data computations in business intelligence***.

Perceptron model is also treated as one of the best and simplest types of Artificial Neural networks. However, it is a supervised learning algorithm of binary classifiers. Hence, we can consider it as a single-layer neural network with four main parameters, i.e., **input values, weights and Bias, net sum, and an activation function.**

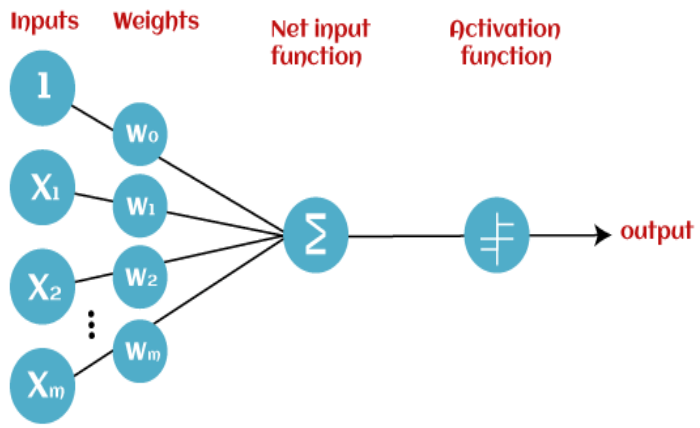
What is Binary classifier in Machine Learning?

In Machine Learning, binary classifiers are defined as the function that helps in deciding whether input data can be represented as vectors of numbers and belongs to some specific class.

Binary classifiers can be considered as linear classifiers. In simple words, we can understand it as a ***classification algorithm that can predict linear predictor function in terms of weight and feature vectors.***

Basic Components of Perceptron

Mr. Frank Rosenblatt invented the perceptron model as a binary classifier which contains three main components. These are as follows:



- **Input Nodes or Input Layer:**

This is the primary component of Perceptron which accepts the initial data into the system for further processing. Each input node contains a real numerical value.

- **Wight and Bias:**

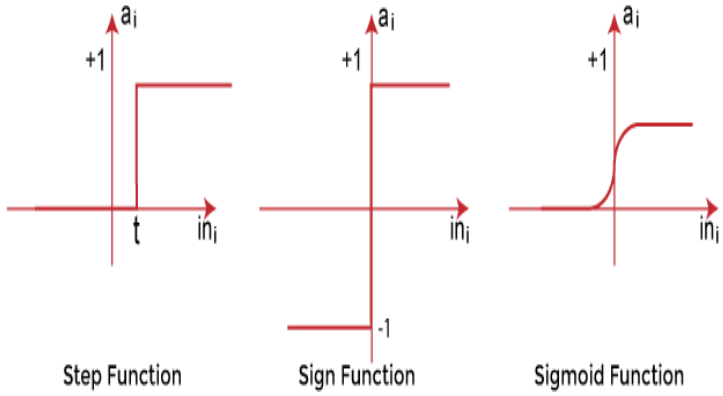
Weight parameter represents the strength of the connection between units. This is another most important parameter of Perceptron components. Weight is directly proportional to the strength of the associated input neuron in deciding the output. Further, Bias can be considered as the line of intercept in a linear equation.

- **Activation Function:**

These are the final and important components that help to determine whether the neuron will fire or not. Activation Function can be considered primarily as a step function.

Types of Activation functions:

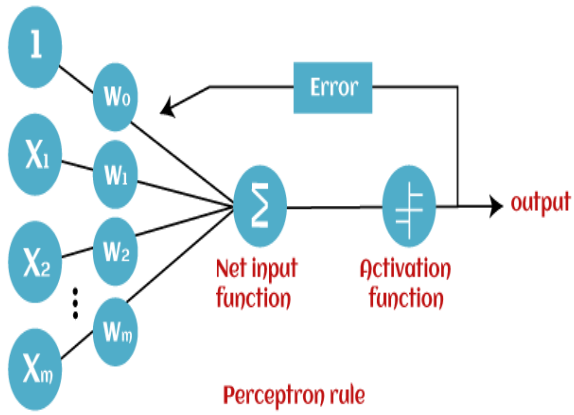
- Sign function
- Step function, and
- Sigmoid function



The data scientist uses the activation function to take a subjective decision based on various problem statements and forms the desired outputs. Activation function may differ (e.g., Sign, Step, and Sigmoid) in perceptron models by checking whether the learning process is slow or has vanishing or exploding gradients.

How does Perceptron work?

In Machine Learning, Perceptron is considered as a single-layer neural network that consists of four main parameters named input values (Input nodes), weights and Bias, net sum, and an activation function. The perceptron model begins with the multiplication of all input values and their weights, then adds these values together to create the weighted sum. Then this weighted sum is applied to the activation function 'f' to obtain the desired output. This activation function is also known as the **step function** and is represented by **'f'**.



This step function or Activation function plays a vital role in ensuring that output is mapped between required values (0,1) or (-1,1). It is important to note that the weight of input is indicative of the strength of a node. Similarly, an input's bias value gives the ability to shift the activation function curve up or down.

Perceptron model works in two important steps as follows:

Step-1

In the first step first, multiply all input values with corresponding weight values and then add them to determine the weighted sum. Mathematically, we can calculate the weighted sum as follows:

$\sum wi*xi = x1*w1 + x2*w2 + \ldots wn*xn$

Add a special term called **bias 'b'** to this weighted sum to improve the model's performance.

$\sum \mathbf{wi*xi + b}$

Step-2

In the second step, an activation function is applied with the above-mentioned weighted sum, which gives us output either in binary form or a continuous value as follows:

$\mathbf{Y = f(\sum wi*xi + b)}$

Types of Perceptron Models

Based on the layers, Perceptron models are divided into two types. These are as follows:

1. Single-layer Perceptron Model
2. Multi-layer Perceptron model

Single Layer Perceptron Model:

This is one of the easiest Artificial neural networks (ANN) types. A single-layered perceptron model consists feed-forward network and also includes a threshold transfer function inside the model. The main objective of the single-layer perceptron model is to analyze the linearly separable objects with binary outcomes.

In a single layer perceptron model, its algorithms do not contain recorded data, so it begins with inconstantly allocated input for weight parameters. Further, it sums up all inputs (weight). After adding all inputs, if the total sum of all inputs is more than a pre-determined value, the model gets activated and shows the output value as +1.

If the outcome is same as pre-determined or threshold value, then the performance of this model is stated as satisfied, and weight demand does not change. However, this model consists of a few discrepancies triggered when multiple weight inputs values are fed into the model. Hence, to find desired output and minimize errors, some changes should be necessary for the weights input.

"Single-layer perceptron can learn only linearly separable patterns."

Multi-Layered Perceptron Model:

Like a single-layer perceptron model, a multi-layer perceptron model also has the same model structure but has a greater number of hidden layers.

The multi-layer perceptron model is also known as the Backpropagation algorithm, which executes in two stages as follows:

- **Forward Stage:** Activation functions start from the input layer in the forward stage and terminate on the output layer.
- **Backward Stage:** In the backward stage, weight and bias values are modified as per the model's requirement. In this stage, the error between actual output and demanded originated backward on the output layer and ended on the input layer.

Hence, a multi-layered perceptron model has considered as multiple artificial neural networks having various layers in which activation function does not remain linear, similar to a single layer perceptron model. Instead of linear, activation function can be executed as sigmoid, TanH, ReLU, etc., for deployment.

A multi-layer perceptron model has greater processing power and can process linear and non-linear patterns. Further, it can also implement logic gates such as AND, OR, XOR, NAND, NOT, XNOR, NOR.

Advantages of Multi-Layer Perceptron:

- A multi-layered perceptron model can be used to solve complex non-linear problems.
- It works well with both small and large input data.
- It helps us to obtain quick predictions after the training.
- It helps to obtain the same accuracy ratio with large as well as small data.

Disadvantages of Multi-Layer Perceptron:

- In Multi-layer perceptron, computations are difficult and time-consuming.
- In multi-layer Perceptron, it is difficult to predict how much the dependent variable affects each independent variable.
- The model functioning depends on the quality of the training.

Perceptron Function

Perceptron function "f(x)" can be achieved as output by multiplying the input 'x' with the learned weight coefficient 'w'.

Mathematically, we can express it as follows:

f(x)=1; if w.x+b>0

otherwise, f(x)=0

- 'w' represents real-valued weights vector
- 'b' represents the bias
- 'x' represents a vector of input x values.

Characteristics of Perceptron

The perceptron model has the following characteristics.

1. Perceptron is a machine learning algorithm for supervised learning of binary classifiers.
2. In Perceptron, the weight coefficient is automatically learned.
3. Initially, weights are multiplied with input features, and the decision is made whether the neuron is fired or not.
4. The activation function applies a step rule to check whether the weight function is greater than zero.
5. The linear decision boundary is drawn, enabling the distinction between the two linearly separable classes +1 and -1.
6. If the added sum of all input values is more than the threshold value, it must have an output signal; otherwise, no output will be shown.

Limitations of Perceptron Model

A perceptron model has limitations as follows:

- The output of a perceptron can only be a binary number (0 or 1) due to the hard limit transfer function.
- Perceptron can only be used to classify the linearly separable sets of input vectors. If input vectors are non-linear, it is not easy to classify them properly.

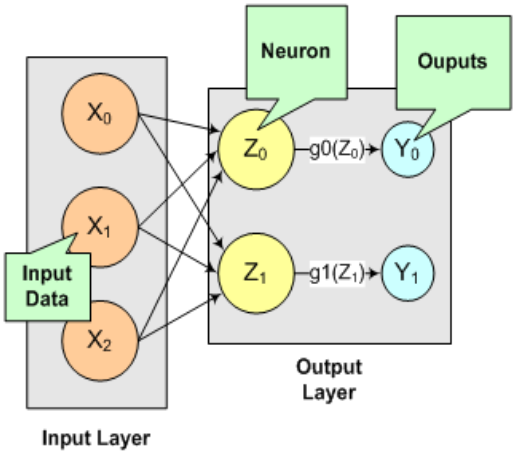
Future of Perceptron

The future of the Perceptron model is much bright and significant as it helps to interpret data by building intuitive patterns and applying them in the future. Machine learning is a rapidly growing

technology of Artificial Intelligence that is continuously evolving and in the developing phase; hence the future of perceptron technology will continue to support and facilitate analytical behavior in machines that will, in turn, add to the efficiency of computers.
The perceptron model is continuously becoming more advanced and working efficiently on complex problems with the help of artificial neurons.

Multi-Layer Feed Forward Networks

A multilayer feed forward neural network is an interconnection of perceptrons in which data and calculations flow in a single direction, from the input data to the outputs. The number of layers in a neural network is the number of layers of perceptrons. The simplest neural network is one with a single input layer and an output layer of perceptrons. The network in Figure 13-7 illustrates this type of network. Technically, this is referred to as a one-layer feedforward network with two outputs because the output layer is the only layer with an activation calculation.



A feed forward neural network is a type of artificial neural network in which nodes' connections do not form a loop.
Often referred to as a multi-layered network of neurons, feed forward neural networks are so named because all information flows in a forward manner only.
The data enters the input nodes, travels through the hidden layers, and eventually exits the output nodes. The network is devoid of links that would allow the information exiting the output node to be sent back into the network.
The purpose of feed forward neural networks is to approximate functions.
Here's how it works
There is a classifier using the formula y = f* (x).
This assigns the value of input x to the category y.
The feed forward network will map y = f (x; θ). It then memorizes the value of θ that most closely approximates the function.
As shown in the Google Photos app, a feed forward neural network serves as the foundation for object detection in photos.
A Feedforward Neural Network's Layers

The following are the components of a feedforward neural network:

Layer of input

It contains the neurons that receive input. The data is subsequently passed on to the next tier. The input layer's total number of neurons is equal to the number of variables in the dataset.

Hidden layer

This is the intermediate layer, which is concealed between the input and output layers. This layer has a large number of neurons that perform alterations on the inputs. They then communicate with the output layer.

Output layer

It is the last layer and is depending on the model's construction. Additionally, the output layer is the expected feature, as you are aware of the desired outcome.

Neurons weights

Weights are used to describe the strength of a connection between neurons. The range of a weight's value is from 0 to 1.

Cost Function in Feedforward Neural Network

The cost function is an important factor of a feedforward neural network. Generally, minor adjustments to weights and biases have little effect on the categorized data points. Thus, to determine a method for improving performance by making minor adjustments to weights and biases using a smooth cost function.

The mean square error cost function is defined as follows:

$$C(w, b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2.$$

Where,

w = weights collected in the network

b = biases

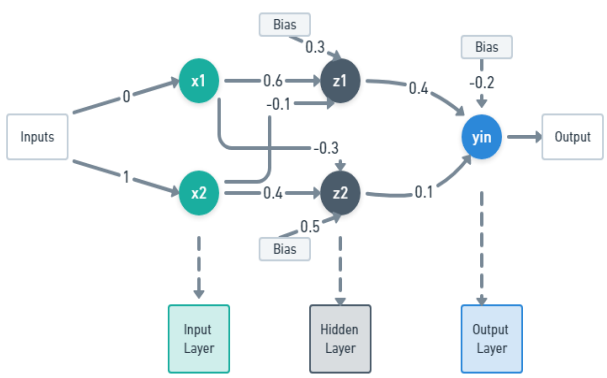
n = number of training inputs

a = output vectors

x = input

‖v‖ = usual length of vector v

Example:



The network in the above figure is a simple multi-layer feed-forward network or backpropagation network. It contains three layers, the input layer with two neurons x_1 and x_2, the hidden layer with two neurons z_1 and z_2 and the output layer with one neuron y_{in}.

Now let's write down the weights and bias vectors for each neuron.

Note: The weights are taken randomly.

Input layer: i/p – $[x_1\ x_2] = [0\ 1]$

Here since it is the input layer only the input values are present.

Hidden layer: $z_1 – [v_{11}\ v_{21}\ v_{01}] = [0.6\ -0.1\ 03]$

Here v_{11} refers to the weight of first input x_1 on z_1, v_{21} refers to the weight of second input x_2 on z_1 and v_{01} refers to the bias value on z_1.

$z_2 – [v_{12}\ v_{22}\ v_{02}] = [-0.3\ 0.4\ 0.5]$

Here v_{12} refers to the weight of first input x_1 on z_2, v_{22} refers to the weight of second input x_2 on z_2 and v_{02} refers to the bias value on z_2.

Output layer: $y_{in} – [w_{11}\ w_{21}\ w_{01}] = [0.4\ 0.1\ -0.2]$

Here w_{11} refers to the weight of first neuron z_1 in a hidden layer on y_{in}, w_{21} refers to the weight of second neuron z_2 in a hidden layer on y_{in} and w_{01} refers to the bias value on y_{in}. Let's consider three variables, k which refers to the neurons in the output layer, 'j' which refers to the neurons in the hidden layer and 'i' which refers to the neurons in the input layer.

Therefore,

$k = 1$
$j = 1, 2$(meaning first neuron and second neuron in hidden layer)
$i = 1, 2$(meaning first and second neuron in the input layer)
Below are some conditions to be followed in BPNs.

Conditions/Constraints:

1. In BPN, the activation function used should be differentiable.
2. The input for bias is always 1.

To proceed with the problem, let:
Target value, $t = 1$
Learning rate, $\alpha = 0.25$
Activation function = Binary sigmoid function
Binary sigmoid function, $f(x) = (1+e^{-x})^{-1}$ eq. (1)
And, $f'(x) = f(x)[1-f(x)]$ eq. (2)
There are three steps to solve the problem:

1. Computing the output, y.
2. Backpropagation of errors, i.e., between output and hidden layer, hidden and input layer.
3. Updating weights.

Step 1:
The value y is calculated by finding y_{in} and applying the activation function.
y_{in} is calculated as:
$y_{in} = w_{01} + z_1*w_{11} + z_2*w_{21}$ eq. (3)
Here, z_1 and z_2 are the values from hidden layer, calculated by finding z_{in1}, z_{in2} and applying activation function to them.
z_{in1} and z_{in2} are calculated as:
$z_{in1} = v_{01} + x_1*v_{11} + x_2*v_{21}$ eq. (4)
$z_{in2} = v_{02} + x_1*v_{12} + x_2*v_{22}$ eq. (5)
From (4)
$z_{in1} = 0.3 + 0*0.6 + 1*(-0.1)$
$z_{in1} = 0.2$
$z_1 = f(z_{in1}) = (1+e^{-0.2})^{-1}$ From (1)
$\mathbf{z_1 = 0.5498}$
From (5)
$z_{in2} = 0.5 + 0*(-0.3) + 1*0.4$
$z_{in2} = 0.9$
$z_2 = f(z_{in2}) = (1+e^{-0.9})^{-1}$ From (1)
$\mathbf{z_2 = 0.7109}$
From (3)
$y_{in} = (-0.2) + 0.5498*0.4 + 0.7109*0.1$
$y_{in} = 0.0910$
$y = f(y_{in}) = (1+e^{-0.0910})^{-1}$ From (1)
$\mathbf{y = 0.5227}$
Here, y is not equal to the target 't', which is 1. And we proceed to calculate the errors and then update weights from them in order to achieve the target value.

Step 2:

(a) Calculating the error between output and hidden layer

Error between output and hidden layer is represented as δ_k, where k represents the neurons in output layer as mentioned above. The error is calculated as:
$\delta_k = (t_k - y_k) * f'(y_{ink})$ eq. (6)
where, $f'(y_{ink}) = f(y_{ink})[1 - f(y_{ink})]$ From (2)
Since $k = 1$ (Assumed above),
$\delta = (t - y)\, f'(y_{in})$ eq. (7)
where, $f'(y_{in}) = f(y_{in})[1 - f(y_{in})]$
$f'(y_{in}) = 0.5227[1 - 0.5227]$

$f'(y_{in}) = 0.2495$
Therefore,
$\delta = (1 – 0.5227) * 0.2495$ From (7)
δ = 0.1191, is the error
Note: (Target – Output) i.e., (t – y) is the error in the output not in the layer. Error in a layer is contributed by different factors like weights and bias.
(b) Calculating the error between hidden and input layer
Error between hidden and input layer is represented as δ_j, where j represents the number of neurons in the hidden layer as mentioned above. The error is calculated as:
$\delta_j = \delta_{inj} * f'(z_{inj})$ eq. (8)
where,
$\delta_{inj} = \sum_{k=1 \text{ to } n} (\delta_k * w_{jk})$ eq. (9)
$f'(z_{inj}) = f(z_{inj})[1 – f(z_{inj})]$ eq. (10)
Since k = 1(Assumed above) eq. (9) becomes:
$\delta_{inj} = \delta * w_{j1}$ eq. (11)
As j = 1, 2, we will have one error values for each neuron and total of 2 errors values.
$\delta_1 = \delta_{in1} * f'(z_{in1})$ eq. (12), From (8)
$\delta_{in1} = \delta * w_{11}$ From (11)
$\delta_{in1} = 0.1191 * 0.4$ From weights vectors
$\delta_{in1} = 0.04764$
$f'(z_{in1}) = f(z_{in1})[1 – f(z_{in1})]$
$f'(z_{in1}) = 0.5498[1 – 0.5498]$ As $f(z_{in1}) = z_1$
$f'(z_{in1}) = 0.2475$
Substituting in (12)
$\delta_1 = 0.04674 * 0.2475 = 0.0118$
$\delta_2 = \delta_{in2} * f'(z_{in2})$ eq. (13), From (8)
$\delta_{in2} = \delta * w_{21}$ From (11)
$\delta_{in2} = 0.1191 * 0.1$ From weights vectors
$\delta_{in2} = 0.0119$
$f'(z_{in2}) = f(z_{in2})[1 – f(z_{in2})]$
$f'(z_{in2}) = 0.7109[1 – 0.7109]$ As $f(z_{in2}) = z_2$
$f'(z_{in2}) = 0.2055$
Substituting in (13)
$\delta_2 = 0.0119 * 0.2055 = 0.00245$
The errors have been calculated, the weights have to be updated using these error values.
Step 3:
The formula for updating weights for output layer is:
$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk}$ eq. (14)
where, $\Delta w_{jk} = \alpha * \delta_k * z_j$ eq. (15)
Since k = 1, (15) becomes:
$\Delta w_{jk} = \alpha * \delta * z_i$ eq. (16)
The formula for updating weights for hidden layer is:
$v_{ij}(new) = v_{ij}(old) + \Delta v_{ij}$ eq. (17)
where, $\Delta v_i = \alpha * \delta_j * x_i$ eq. (18)
From (14) and (16)
$w_{11}(new) = w_{11}(old) + \Delta w_{11} = 0.4 + \alpha * \delta * z_1 = 0.4 + 0.25 * 0.1191 * 0.5498 = 0.4164$
$w_{21}(new) = w_{21}(old) + \Delta w_{21} = 0.1 + \alpha * \delta * z_2 = 0.1 + 0.25 * 0.1191 * 0.7109 = 0.12117$
$w_{01}(new) = w_{01}(old) + \Delta w_{01} = (-0.2) + \alpha * \delta * bias = (-0.2) + 0.25 * 0.1191 * 1 = -0.1709$, kindly note the 1 taken here is input considered for bias as per the conditions.
These are the updated weights of the output layer.
From (17) and (18)
$v_{11}(new) = v_{11}(old) + \Delta v_{11} = 0.6 + \alpha * \delta_1 * x_1 = 0.6 + 0.25 * 0.0118 * 0 = 0.6$
$v_{21}(new) = v_{21}(old) + \Delta v_{21} = (-0.1) + \alpha * \delta_1 * x_2 = (-0.1) + 0.25 * 0.0118 * 1 = 0.00295$

$v_{01}(new) = v_{01}(old) + \Delta v_{01} = 0.3 + \alpha * \delta_1 * bias = 0.3 + 0.25 * 0.0118 * 1 = 0.00295$, kindly note the 1 taken here is input considered for bias as per the conditions.
$v_{12}(new) = v_{12}(old) + \Delta v_{12} = (-0.3) + \alpha * \delta_2 * x_1 = (-0.3) + 0.25 * 0.00245 * 0 = -0.3$
$v_{22}(new) = v_{22}(old) + \Delta v_{22} = 0.4 + \alpha * \delta_2 * x_2 = 0.4 + 0.25 * 0.00245 * 1 = 0.400612$
$v_{02}(new) = v_{02}(old) + \Delta v_{02} = 0.5 + \alpha * \delta_2 * bias = 0.5 + 0.25 * 0.00245 * 1 = 0.500612$, kindly note the 1 taken here is input considered for bias as per the conditions.
These are all the updated weights of the hidden layer.
These three steps are repeated until the output 'y' is equal to the target 't'.
This is how the BPNs work. The backpropagation in BPN refers to that the error in the present layer is used to update weights between the present and previous layer by backpropagating the error values.

Bayes' theorem
Bayes' theorem is also known as **Bayes' rule, Bayes' law**, or **Bayesian reasoning**, which determines the probability of an event with uncertain knowledge.
In probability theory, it relates the conditional probability and marginal probabilities of two random events.
Bayes' theorem was named after the British mathematician **Thomas Bayes**. The **Bayesian inference** is an application of Bayes' theorem, which is fundamental to Bayesian statistics.
It is a way to calculate the value of P(B|A) with the knowledge of P(A|B).
Bayes' theorem allows updating the probability prediction of an event by observing new information of the real world.
Example: If cancer corresponds to one's age then by using Bayes' theorem, we can determine the probability of cancer more accurately with the help of age.
Bayes' theorem can be derived using product rule and conditional probability of event A with known event B:
As from product rule we can write:
$P(A \wedge B) = P(A|B)\, P(B)$ or
Similarly, the probability of event B with known event A:
$P(A \wedge B) = P(B|A)\, P(A)$
Equating right hand side of both the equations, we will get:

$$P(A|B) = \frac{P(B|A)\, P(A)}{P(B)} \quad \text{....(a)}$$

The above equation (a) is called as **Bayes' rule** or **Bayes' theorem**. This equation is basic of most modern AI systems for **probabilistic inference**.
It shows the simple relationship between joint and conditional probabilities. Here,
P(A|B) is known as **posterior**, which we need to calculate, and it will be read as Probability of hypothesis A when we have occurred an evidence B.
P(B|A) is called the likelihood, in which we consider that hypothesis is true, then we calculate the probability of evidence.
P(A) is called the **prior probability**, probability of hypothesis before considering the evidence
P(B) is called **marginal probability**, pure probability of an evidence.
In the equation (a), in general, we can write P (B) = P(A)*P(B|Ai), hence the Bayes' rule can be written as:

$$P(A_i|B) = \frac{P(A_i) * P(B|A_i)}{\sum_{i=1}^{k} P(A_i) * P(B|A_i)}$$

Where $A_1, A_2, A_3, \ldots\ldots, A_n$ is a set of mutually exclusive and exhaustive events.
Applying Bayes' rule:
Bayes' rule allows us to compute the single term P(B|A) in terms of P(A|B), P(*B*), and P(A). This is very useful in cases where we have a good probability of these three terms and want to determine the

fourth one. Suppose we want to perceive the effect of some unknown cause, and want to compute that cause, then the Bayes' rule becomes:

$$P(\text{cause}|\text{effect}) = \frac{P(\text{effect}|\text{cause})\,P(\text{cause})}{P(\text{effect})}$$

Example-1:
Question: what is the probability that a patient has diseases meningitis with a stiff neck?
Given Data:
A doctor is aware that disease meningitis causes a patient to have a stiff neck, and it occurs 80% of the time. He is also aware of some more facts, which are given as follows:

- The Known probability that a patient has meningitis disease is 1/30,000.
- The Known probability that a patient has a stiff neck is 2%.

Let a be the proposition that patient has stiff neck and b be the proposition that patient has meningitis. , so we can calculate the following as:
P(a|b) = 0.8
P(b) = 1/30000
P(a)= .02

$$P(b|a) = \frac{P(a|b)P(b)}{P(a)} = \frac{0.8*(\frac{1}{30000})}{0.02} = 0.001333333.$$

Hence, we can assume that 1 patient out of 750 patients has meningitis disease with a stiff neck.
Example-2:
Question: From a standard deck of playing cards, a single card is drawn. The probability that the card is king is 4/52, then calculate posterior probability P(King|Face), which means the drawn face card is a king card.
Solution:

$$P(\text{king}|\text{face}) = \frac{P(\text{Face}|\text{king})*P(\text{King})}{P(\text{Face})} \quad .....(i)$$

P(king): probability that the card is King= 4/52= 1/13
P(face): probability that a card is a face card= 3/13
P(Face|King): probability of face card when we assume it is a king = 1
Putting all values in equation (i) we will get:

$$P(\text{king}|\text{face}) = \frac{1*(\frac{1}{13})}{(\frac{3}{13})} = 1/3, \text{ it is a probability that a face card is a king card.}$$

Following are some applications of Bayes' theorem:

- It is used to calculate the next step of the robot when the already executed step is given.
- Bayes' theorem is helpful in weather forecasting.
- It can solve the Monty Hall problem.

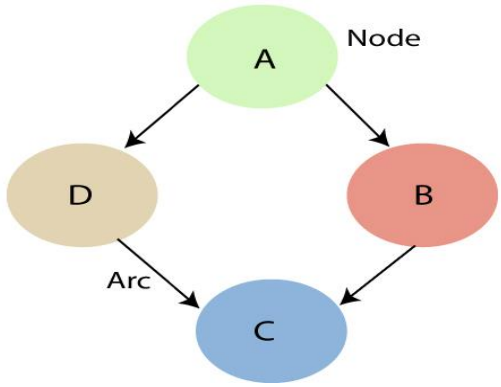
Bayesian Belief Network
Bayesian belief network is key computer technology for dealing with probabilistic events and to solve a problem which has uncertainty. We can define a Bayesian network as:
"A Bayesian network is a probabilistic graphical model which represents a set of variables and their conditional dependencies using a directed acyclic graph."
It is also called a **Bayes network, belief network, decision network**, or **Bayesian model**.
Bayesian networks are probabilistic, because these networks are built from a **probability distribution**, and also use probability theory for prediction and anomaly detection.
Real world applications are probabilistic in nature, and to represent the relationship between multiple events, we need a Bayesian network. It can also be used in various tasks including **prediction, anomaly detection, diagnostics, automated insight, reasoning, time series prediction**, and **decision making under uncertainty**.

Bayesian Network can be used for building models from data and experts opinions, and it consists of two parts:

- **Directed Acyclic Graph**
- **Table of conditional probabilities.**

The generalized form of Bayesian network that represents and solve decision problems under uncertain knowledge is known as an **Influence diagram**.

A Bayesian network graph is made up of nodes and Arcs (directed links), where:



- Each **node** corresponds to the random variables, and a variable can be **continuous** or **discrete**.
- **Arc or directed arrows** represent the causal relationship or conditional probabilities between random variables. These directed links or arrows connect the pair of nodes in the graph. These links represent that one node directly influence the other node, and if there is no directed link that means that nodes are independent with each other
    - **In the above diagram, A, B, C, and D are random variables represented by the nodes of the network graph.**
    - **If we are considering node B, which is connected with node A by a directed arrow, then node A is called the parent of Node B.**
    - **Node C is independent of node A.**

The Bayesian network has mainly two components:

- **Causal Component**
- **Actual numbers**

Each node in the Bayesian network has condition probability distribution $\mathbf{P(X_i \mid Parent(X_i))}$, which determines the effect of the parent on that node.

Bayesian network is based on Joint probability distribution and conditional probability. So let's first understand the joint probability distribution:

Joint probability distribution:

If we have variables x1, x2, x3,....., xn, then the probabilities of a different combination of x1, x2, x3.. xn, are known as Joint probability distribution.

$\mathbf{P[x_1, x_2, x_3,....., x_n]}$, it can be written as the following way in terms of the joint probability distribution.

$\mathbf{= P[x_1 \mid x_2, x_3,....., x_n]P[x_2, x_3,....., x_n]}$

$\mathbf{= P[x_1 \mid x_2, x_3,....., x_n]P[x_2 \mid x_3,....., x_n]....P[x_{n-1} \mid x_n]P[x_n].}$

In general for each variable Xi, we can write the equation as:

$P(X_i \mid X_{i-1},....., X_1) = P(X_i \mid Parents(X_i))$

Explanation of Bayesian network:

Let's understand the Bayesian network through an example by creating a directed acyclic graph:

Example: Harry installed a new burglar alarm at his home to detect burglary. The alarm reliably responds at detecting a burglary but also responds for minor earthquakes. Harry has two neighbors David and Sophia, who have taken a responsibility to inform Harry at work when they hear the alarm. David always calls Harry when he hears the alarm, but sometimes he got confused with the phone ringing and calls at that time too. On the other hand, Sophia likes to listen to high music, so sometimes she misses to hear the alarm. Here we would like to compute the probability of Burglary Alarm.

Problem:

Calculate the probability that alarm has sounded, but there is neither a burglary, nor an earthquake occurred, and David and Sophia both called the Harry.

Solution:

- The Bayesian network for the above problem is given below. The network structure is showing that burglary and earthquake is the parent node of the alarm and directly affecting the probability of alarm's going off, but David and Sophia's calls depend on alarm probability.
- The network is representing that our assumptions do not directly perceive the burglary and also do not notice the minor earthquake, and they also not confer before calling.
- The conditional distributions for each node are given as conditional probabilities table or CPT.
- Each row in the CPT must be sum to 1 because all the entries in the table represent an exhaustive set of cases for the variable.
- In CPT, a boolean variable with k boolean parents contains 2^K probabilities. Hence, if there are two parents, then CPT will contain 4 probability values

List of all events occurring in this network:

- **Burglary (B)**
- **Earthquake(E)**
- **Alarm(A)**
- **David Calls(D)**
- **Sophia calls(S)**

We can write the events of problem statement in the form of probability: **P[D, S, A, B, E]**, can rewrite the above probability statement using joint probability distribution:

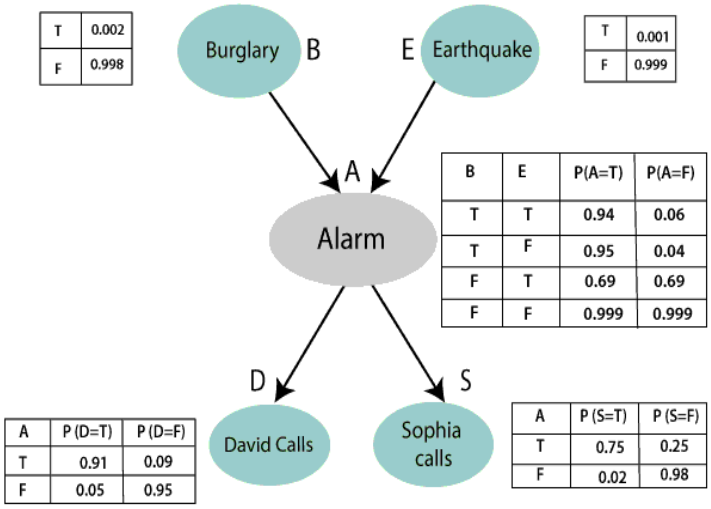
P[D, S, A, B, E]= P[D | S, A, B, E]. P[S, A, B, E]

=P[D | S, A, B, E]. P[S | A, B, E]. P[A, B, E]

= P [D| A]. P [S| A, B, E]. P[A, B, E]

= P[D | A]. P[S | A]. P[A| B, E]. P[B, E]

= P[D | A]. P[S | A]. P[A| B, E]. P[B |E]. P[E]



T	0.002
F	0.998

T	0.001
F	0.999

B	E	P(A=T)	P(A=F)
T	T	0.94	0.06
T	F	0.95	0.04
F	T	0.69	0.69
F	F	0.999	0.999

A	P (D=T)	P (D=F)
T	0.91	0.09
F	0.05	0.95

A	P (S=T)	P (S=F)
T	0.75	0.25
F	0.02	0.98

Let's take the observed probability for the Burglary and earthquake component:

P(B= True) = 0.002, which is the probability of burglary.

P(B= False)= 0.998, which is the probability of no burglary.

P(E= True)= 0.001, which is the probability of a minor earthquake

P(E= False)= 0.999, Which is the probability that an earthquake not occurred.

We can provide the conditional probabilities as per the below tables:

Conditional probability table for Alarm A:

The Conditional probability of Alarm A depends on Burglar and earthquake:

B	E	P(A= True)	P(A= False)
True	True	0.94	0.06
True	False	0.95	0.04
False	True	0.31	0.69
False	False	0.001	0.999

Conditional probability table for David Calls:
The Conditional probability of David that he will call depends on the probability of Alarm.

A	P(D= True)	P(D= False)
True	0.91	0.09
False	0.05	0.95

Conditional probability table for Sophia Calls:
The Conditional probability of Sophia that she calls is depending on its Parent Node "Alarm."

A	P(S= True)	P(S= False)
True	0.75	0.25
False	0.02	0.98

From the formula of joint distribution, we can write the problem statement in the form of probability distribution:
P(S, D, A, ¬B, ¬E) = P (S|A) *P (D|A)*P (A|¬B ^ ¬E) *P (¬B) *P (¬E).
= 0.75* 0.91* 0.001* 0.998*0.999
= 0.00068045.
Hence, a Bayesian network can answer any query about the domain by using Joint distribution.