Unit-1

Basic Concepts in Machine Learning

Learning Algorithms

Machine Learning algorithms are the programs that can learn the hidden patterns from the data, predict the output, and improve the performance from experiences on their own. Different algorithms can be used in machine learning for different tasks, such as simple linear regression that can be used **for prediction problem**s like **stock market prediction,** and **the KNN algorithm can be used for classification problems**.

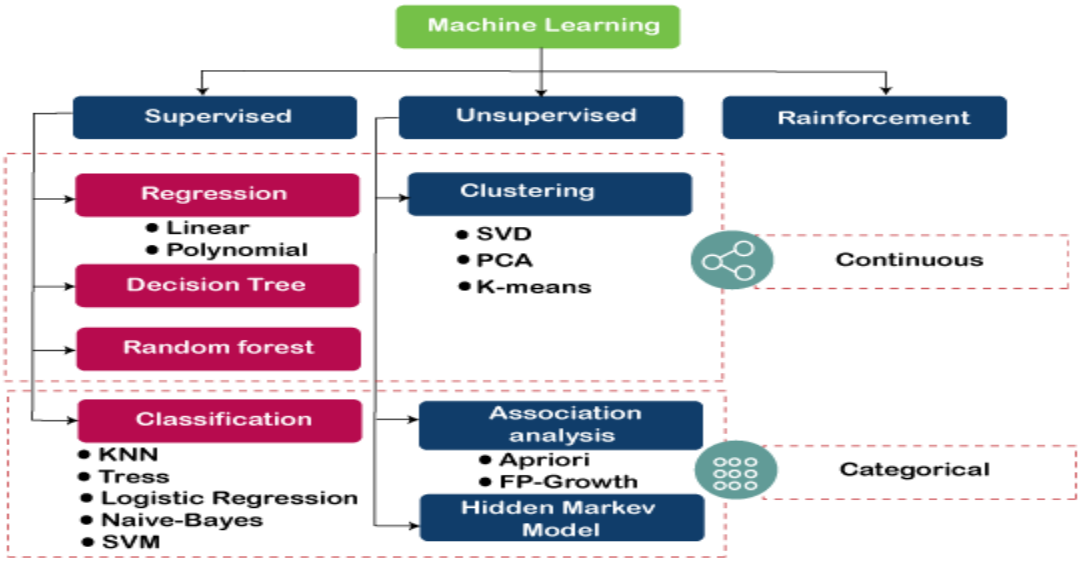
Lets discuss the overview of some popular and most commonly used machine learning algorithms along with their use cases and categories.

Types of Machine Learning Algorithms

Machine Learning Algorithm can be broadly classified into three types:

1. **Supervised Learning Algorithms**
2. **Unsupervised Learning Algorithms**
3. **Reinforcement Learning algorithm**

The below diagram illustrates the different ML algorithm, along with the categories:



1) Supervised Learning Algorithm

Supervised learning is a type of Machine learning in which the machine needs external supervision to learn. The supervised learning models are trained using the labeled dataset. Once the training and processing are done, the model is tested by providing a sample test data to check whether it predicts the correct output.

The goal of supervised learning is to map input data with the output data. Supervised learning is based on supervision, and it is the same as when a student learns things in the teacher's supervision. The example of supervised learning is **spam filtering**.

Supervised learning can be divided further into two categories of problem:

- Classification
- Regression

Examples of some popular supervised learning algorithms are Simple Linear regression, Decision Tree, Logistic Regression, KNN algorithm, etc.

2) Unsupervised Learning Algorithm

It is a type of machine learning in which the machine does not need any external supervision to learn from the data, hence called unsupervised learning. The unsupervised models can be trained using the unlabelled dataset that is not classified, nor categorized, and the algorithm needs to act on that data without any supervision. In unsupervised learning, the model doesn't have a predefined output, and it tries to find useful insights from the huge amount of data. These are used to solve the Association and Clustering problems**. Hence further, it can be classified into two types:**

- Clustering
- Association

Examples of some Unsupervised learning algorithms are **K-means Clustering, Apriori Algorithm, Eclat, etc.**

3) Reinforcement Learning

In Reinforcement learning, an agent interacts with its environment by producing actions, and learn with the help of feedback. The feedback is given to the agent in the form of rewards, such as for each good action, he gets a positive reward, and for each bad action, he gets a negative reward. There is no supervision provided to the agent. **Q-Learning algorithm** is used in reinforcement learning.

List of Popular Machine Learning Algorithm

1. **Linear Regression Algorithm**
2. **Logistic Regression Algorithm**
3. **Decision Tree**
4. **SVM**
5. **Naïve Bayes**
6. **KNN**
7. **K-Means Clustering**

8. **Random Forest**
9. **Apriori**
10. **PCA**

1. Linear Regression

Linear regression is one of the most popular and simple machine learning algorithms that is used for predictive analysis. Here, **predictive analysis** defines prediction of something, and linear regression makes predictions for *continuous numbers* such as **salary, age, etc.**

It shows the linear relationship between the dependent and independent variables, and shows how the dependent variable(y) changes according to the independent variable (x).

It tries to best fit a line between the dependent and independent variables, and this best fit line is knowns as the regression line.

The equation for the regression line is:

$$y= a_0+ a*x+ b$$

Here, y= dependent variable

x= independent variable

a_0 = Intercept of line.

Linear regression is further divided into two types:

- **Simple Linear Regression:** In simple linear regression, a single independent variable is used to predict the value of the dependent variable.
- **Multiple Linear Regression:** In multiple linear regression, more than one independent variables are used to predict the value of the dependent variable.

2. Logistic Regression

Logistic regression is the supervised learning algorithm, which is used to **predict the categorical variables or discrete values**. It can be used for the *classification problems in machine learning*, and the output of the logistic regression algorithm can be either Yes or NO, 0 or 1, Red or Blue, etc.

Logistic regression is similar to the linear regression except how they are used, such as Linear regression is used to solve the regression problem and predict continuous values, whereas Logistic regression is used to solve the Classification problem and used to predict the discrete values.

Instead of fitting the best fit line, it forms an S-shaped curve that lies between 0 and 1. The S-shaped curve is also known as a logistic function that uses the concept of the threshold. Any value above the threshold will tend to 1, and below the threshold will tend to 0.

3. Decision Tree Algorithm

A decision tree is a supervised learning algorithm that is mainly used to solve the classification problems but can also be used for solving the regression problems. It can work with both categorical variables and continuous variables. It shows a tree-like structure that includes nodes and branches, and starts with the root node that expand on further branches till the leaf node. The **internal node** is used to represent the **features of the dataset, branches show the decision rules,** and **leaf nodes represent the outcome of the problem.**

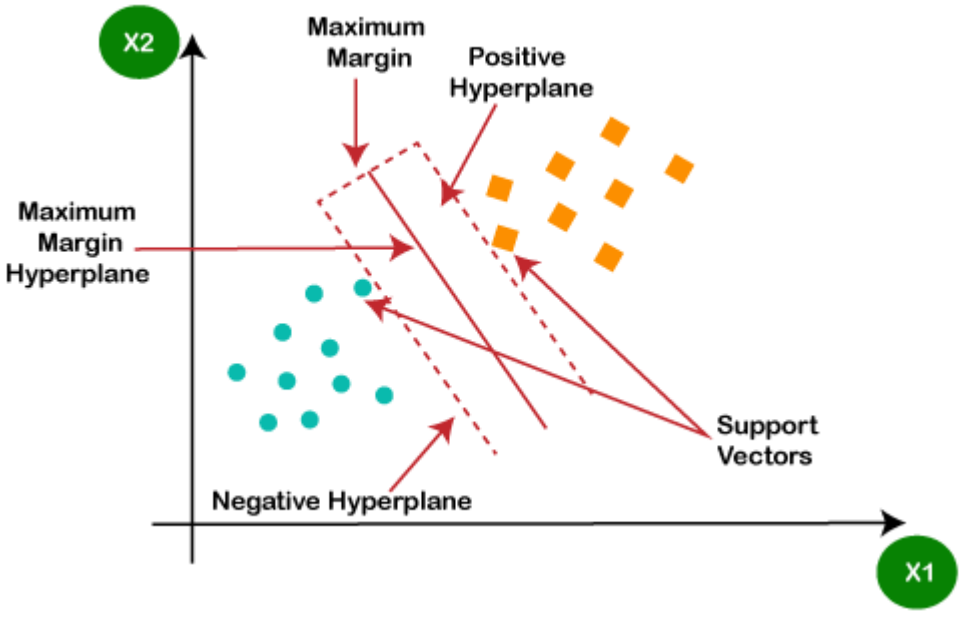
Some real-world applications of decision tree algorithms are identification between cancerous and non-cancerous cells, suggestions to customers to buy a car, etc.

4. Support Vector Machine Algorithm

A support vector machine or SVM is a supervised learning algorithm that can also be used for classification and regression problems. However, it is primarily used for classification problems. The goal of SVM is to create a hyperplane or decision boundary that can segregate datasets into different classes.

The data points that help to define the hyperplane are known as **support vectors**, and hence it is named as support vector machine algorithm.

Some real-life applications of SVM are **face detection, image classification, Drug discovery**, etc. Consider the below diagram:



As we can see in the above diagram, the hyperplane has classified datasets into two different classes.

5. Naïve Bayes Algorithm:

Naïve Bayes classifier is a supervised learning algorithm, which is used to make predictions based on the probability of the object. The algorithm named as Naïve Bayes as it is based on **Bayes theorem**, and follows the *naïve* assumption that says' variables are independent of each other.

The Bayes theorem is based on the conditional probability; it means the likelihood that event(A) will happen, when it is given that event(B) has already happened. The equation for Bayes theorem is given as:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Naïve Bayes classifier is one of the best classifiers that provide a good result for a given problem. It is easy to build a naïve bayesian model, and well suited for the huge amount of dataset. It is mostly used for **text classification**.

6. K-Nearest Neighbour (KNN)

K-Nearest Neighbour is a supervised learning algorithm that can be used for both classification and regression problems. This algorithm works by assuming the similarities between the new data point and available data points. Based on these similarities, the new data points are put in the most similar categories. It is also known as the lazy learner algorithm as it stores all the available datasets and classifies each new case with the help of K-neighbours. The new case is assigned to the nearest class with most similarities, and any distance function measures the distance between the data points. The distance function can be **Euclidean, Minkowski, Manhattan, or Hamming distance**, based on the requirement.

7. K-Means Clustering

K-means clustering is one of the simplest unsupervised learning algorithms, which is used to solve the clustering problems. The datasets are grouped into K different clusters based on similarities and dissimilarities, it means, datasets with most of the commonalties remain in one cluster which has very less or no commonalities between other clusters. In K-means, K-refers to the number of clusters, and **means** refer to the averaging the dataset in order to find the centroid.

It is a centroid-based algorithm, and each cluster is associated with a centroid. This algorithm aims to reduce the distance between the data points and their centroids within a cluster.

This algorithm starts with a group of randomly selected centroids that form the clusters at starting and then perform the iterative process to optimize these centroids' positions.

It can be used for spam detection and filtering, identification of fake news, etc.

8. Random Forest Algorithm

Random forest is the supervised learning algorithm that can be used for both classification and regression problems in machine learning. It is an ensemble learning technique that provides the predictions by combining the multiple classifiers and improve the performance of the model.

It contains multiple decision trees for subsets of the given dataset, and find the average to improve the predictive accuracy of the model. A random-forest should contain 64-128 trees. The greater number of trees leads to higher accuracy of the algorithm.

To classify a new dataset or object, each tree gives the classification result and based on the majority votes, the algorithm predicts the final output.

Random forest is a fast algorithm, and can efficiently deal with the missing & incorrect data.

9. Apriori Algorithm

Apriori algorithm is the unsupervised learning algorithm that is used to solve the association problems. It uses frequent itemsets to generate association rules, and it is designed to work on the databases that contain transactions. With the help of these association rule, it determines how strongly or how weakly two objects are connected to each other. This algorithm uses a breadth-first search and Hash Tree to calculate the itemset efficiently.

The algorithm process iteratively for finding the frequent itemsets from the large dataset.

The apriori algorithm was given by the **R. Agrawal and Srikant** in the year 1994. It is mainly used for market basket analysis and helps to understand the products that can be bought together. It can also be used in the healthcare field to find drug reactions in patients.

10. Principle Component Analysis

Principle Component Analysis (PCA) is an unsupervised learning technique, which is used for dimensionality reduction. It helps in reducing the dimensionality of the dataset that contains many features correlated with each other. It is a statistical process that converts the observations of correlated features into a set of linearly uncorrelated features with the help of orthogonal transformation. It is one of the popular tools that is used for exploratory data analysis and predictive modeling.

PCA works by considering the variance of each attribute because the high variance shows the good split between the classes, and hence it reduces the dimensionality.

Some real-world applications of PCA are image processing, movie recommendation system, optimizing the power allocation in various communication channels.

Maximum Likelihood Estimation

The maximum likelihood estimation is a method that determines values for parameters of the model. It is the statistical method of estimating the parameters of the probability distribution by maximizing the likelihood function. The point in which the parameter value that maximizes the likelihood function is called the maximum likelihood estimate.

Four Major Steps in MLE:

1. Perform a certain experiment to collect the data.
2. Choose a parametric model of the data, with certain modifiable parameters.
3. Formulate the likelihood as an objective function to be maximized.
4. Maximize the objective function and derive the parameters of the model.

A probability distribution for the target variable (labeled class) must be assumed and followed by a likelihood function defined that calculates the probability of observing the outcome given the

input data and the model. The function can be optimized to find the set of parameters that results in the largest sum likelihood over the training dataset.

In Maximum Likelihood Estimation, we maximize the conditional probability of observing the data **(X)** given a specific probability distribution and its parameters **(theta – ɵ)**

- P(X,ɵ) where X is the joint probability distribution of all observations from 1 to n.

P(X1, X2, X3….Xn; ɵ)

- The resulting conditional probability is known as the likelihood of observing the data with the given model parameters and denoted as **(L)**
- L(X, ɵ)

The joint probability can also be defined as the multiplication of the conditional probability for each observation given the distribution parameters

- Sum i to n log [(P(xi, ɵ)]

As log is used mostly in the likelihood function, it is known as log-likelihood function. It is common in optimization problems to prefer to minimize the cost function.

Therefore, the negative of the log-likelihood function is used and known as Negative Log-Likelihood function.

- **Minimize**: Sum i to n log [(P(xi, ɵ)]
- The Maximum Likelihood Estimation framework can be used as a basis for estimating the parameters of many different machine learning models for regression and classification predictive modeling. This includes the logistic regression model.

Let X1, X2, X3, ……, Xn be a random sample from a distribution with a parameter θ. Suppose that we have observed X1=x1, X2=x2, ⋯⋯, Xn=xn.

1. If Xi's are discrete, then the likelihood function is defined as

$L(x1, x2, \cdots, xn; \theta) = Px1x2\cdots xn(x1, x2,\cdots,xn;\theta)$.

1. If Xi's are jointly continuous, then the likelihood function is defined as

$L(x1, x2, \cdots, xn; \theta) = fx1x2\cdots xn(x1, x2,\cdots,xn;\theta)$.

In some problems, it is easier to work with the log likelihood function given by
$\ln L(x1, x2, \cdots, xn ; \theta)$.

Building machine Learning algorithms

The lifecycle of a machine learning project involves a series of steps that include:

1. Study the Problems: The first step is to study the problem. This step involves understanding the business problem and defining the objectives of the model.
2. Data Collection: When the problem is well-defined, we can collect the relevant data required for the model. The data could come from various sources such as databases, APIs, or web scraping.

3. Data Preparation: When our problem-related data is collected. then it is a good idea to check the data properly and make it in the desired format so that it can be used by the model to find the hidden patterns. This can be done in the following steps:
    1. Data cleaning
    2. Data Transformation
    3. Explanatory Data Analysis and Feature Engineering
    4. Split the dataset for training and testing.
4. Model Selection: The next step is to select the appropriate machine learning algorithm that is suitable for our problem. This step requires knowledge of the strengths and weaknesses of different algorithms. Sometimes we use multiple models and compare their results and select the best model as per our requirements.
5. **Model building and Training:** After selecting the algorithm, we have to build the model.

    i. In the case of traditional machine learning building mode is easy it is just a few hyperparameter tunings.
    ii. In the case of deep learning, we have to define layer-wise architecture along with input and output size, number of nodes in each layer, loss function, gradient descent optimizer, etc.
    iii. After that model is trained using the preprocessed dataset.

6. **Model Evaluation:** Once the model is trained, it can be evaluated on the test dataset to determine its accuracy and performance using different techniques like classification report, F1 score, precision, recall, ROC Curve, Mean Square error, absolute error, etc.
7. **Model Tuning:** Based on the evaluation results, the model may need to be tuned or optimized to improve its performance. This involves tweaking the hyperparameters of the model.
8. **Deployment:** Once the model is trained and tuned, it can be deployed in a production environment to make predictions on new data. This step requires integrating the model into an existing software system or creating a new system for the model.
9. **Monitoring and Maintenance:** Finally, it is essential to monitor the model's performance in the production environment and perform maintenance tasks as required. This involves monitoring for data drift, retraining the model as needed, and updating the model as new data becomes available.

Neural Networks Multilayer Perceptron

A multilayer perceptron (MLP) is a feed forward artificial neural network that generates a set of outputs from a set of inputs. An MLP is characterized by several layers of input nodes connected as a directed graph between the input nodes connected as a directed graph between the input and output layers.

Neural Networks are inspired by, but not necessarily an exact model of, the structure of the brain. There's a lot we still don't know about the brain and how it works, but it has been serving as inspiration in many scientific areas due to its ability to develop intelligence. And although there are neural networks that were created with the sole purpose of understanding how brains work, Deep Learning as we know it today is not intended to replicate how the brain works. Instead, Deep Learning focuses on enabling systems that learn multiple levels of pattern composition.

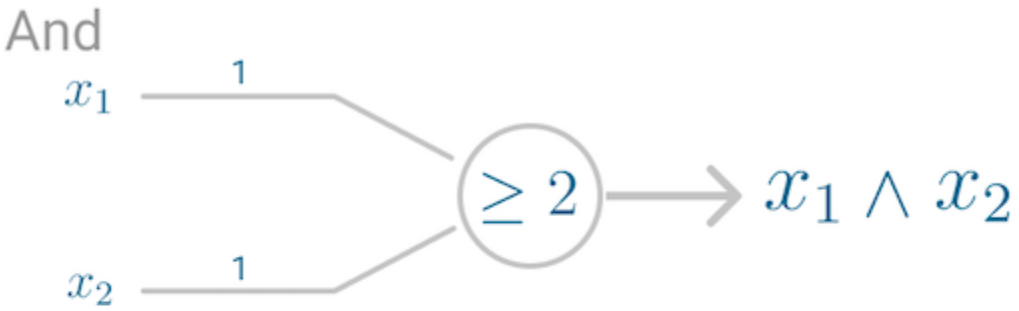
It all started with a basic structure, one that resembles brain's neuron.

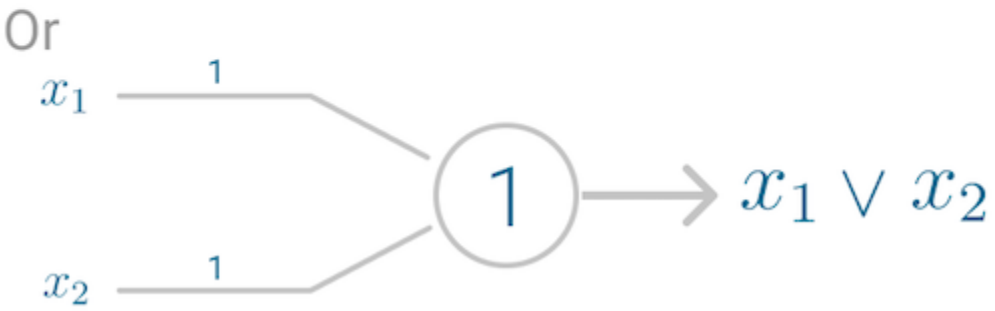
It all started with a Neuron

In the early 1940's **Warren McCulloch**, a neurophysiologist, teamed up with logician Walter Pitts to create a model of how brains work. It was a simple linear model that produced a positive or negative output, given a set of inputs and weights.

$$\underbrace{f(x,w)}_{\text{output}} = \underbrace{x_1 w_1}_{\text{inputs}} + \cdots + x_n \underbrace{w_n}_{\text{weights}}$$

The first application of the neuron replicated a logic gate, where you have one or two binary inputs, and a boolean function that only gets activated given the right inputs and weights.





It was only a decade later that Frank Rosenblatt extended this model, and created an algorithm that could *learn* the weights in order to generate an output.

Building onto McCulloch and Pitt's neuron, Rosenblatt developed the **Perceptron**.

Perceptron

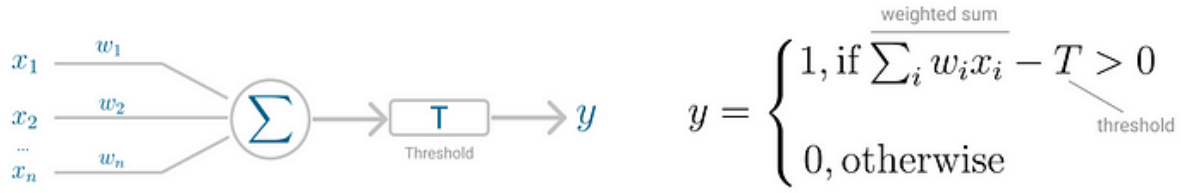
Although today the **Perceptron** is widely recognized as an algorithm, it was initially intended as an image recognition machine. It gets its name from performing the human-like function of *perception*, seeing and recognizing images.

In particular, interest has been centered on the idea of a machine which would be capable of conceptualizing inputs impinging directly from the physical environment of light, sound,

temperature, etc. — the "phenomenal world" with which we are all familiar — rather than requiring the intervention of a human agent to digest and code the necessary information.

Rosenblatt's perceptron machine relied on a basic unit of computation, the **neuron**. Just like in previous models, each neuron has a cell that receives a series of pairs of inputs and weights.

The major difference in Rosenblatt's model is that **inputs are combined in a weighted sum** and, if the weighted sum exceeds a predefined threshold, the neuron fires and produces an output.

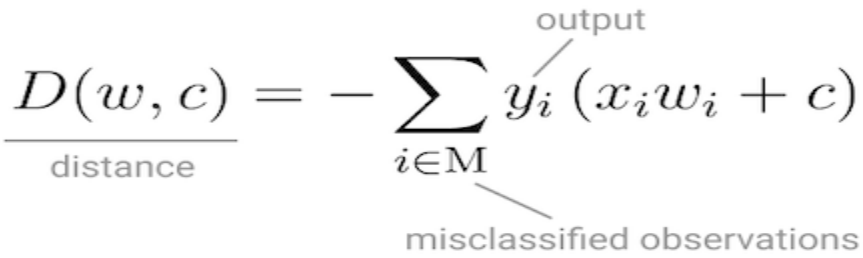


Perceptrons neuron model (left) and threshold logic (right). (Image by author)

Threshold *T* represents the **activation function**. If the weighted sum of the inputs is greater than zero the neuron outputs the value 1, otherwise the output value is zero.

Perceptron for Binary Classification

With this discrete output, controlled by the activation function, the perceptron can be used as a **binary classification model**, defining a **linear decision boundary**. It finds the separating hyperplane that minimizes the distance between misclassified points and the decision boundary.



Perceptron's loss function. (Image by author)

To minimize this distance, Perceptron uses Stochastic Gradient Descent as the optimization function.

If the data is linearly separable, it is guaranteed that Stochastic Gradient Descent will converge in a finite number of steps.

The last piece that Perceptron needs is the **activation function**, the function that determines if the neuron will fire or not.

Initial Perceptron models used sigmoid function, and just by looking at its shape, it makes a lot of sense!

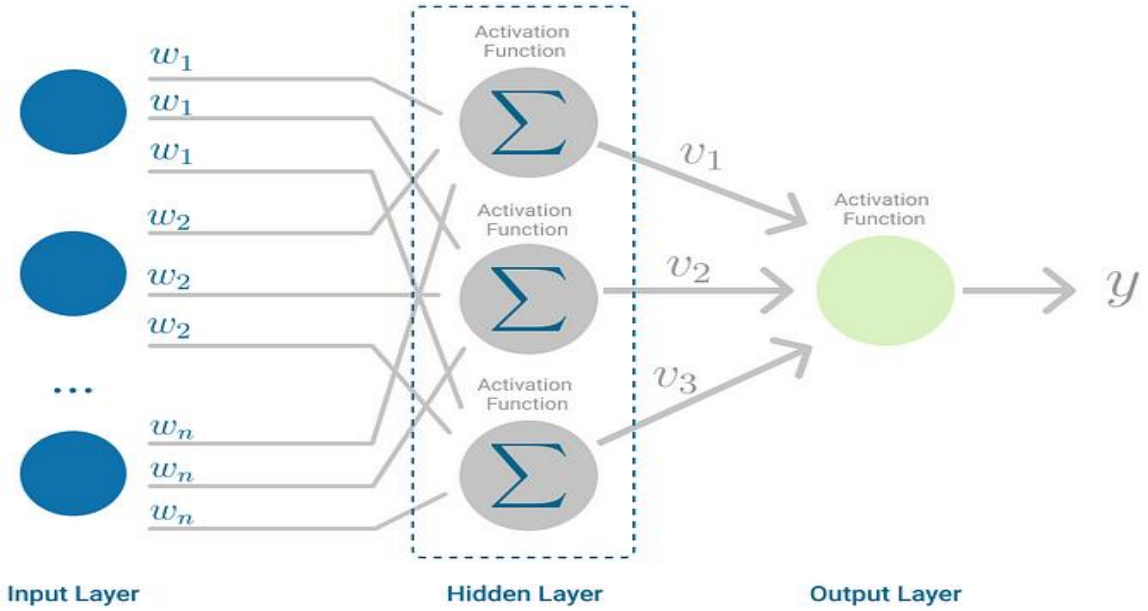
The sigmoid function maps any real input to a value that is either 0 or 1, and encodes a non-linear function.

The neuron can receive negative numbers as input, and it will still be able to produce an output that is either 0 or 1.

Multilayer Perceptron

The **Multilayer Perceptron** was developed to tackle this limitation. It is a neural network where the mapping between inputs and output is non-linear.

A Multilayer Perceptron has input and output layers, and one or more **hidden layers** with many neurons stacked together. And while in the Perceptron the neuron must have an activation function that imposes a threshold, like ReLU or sigmoid, neurons in a Multilayer Perceptron can use any arbitrary activation function.



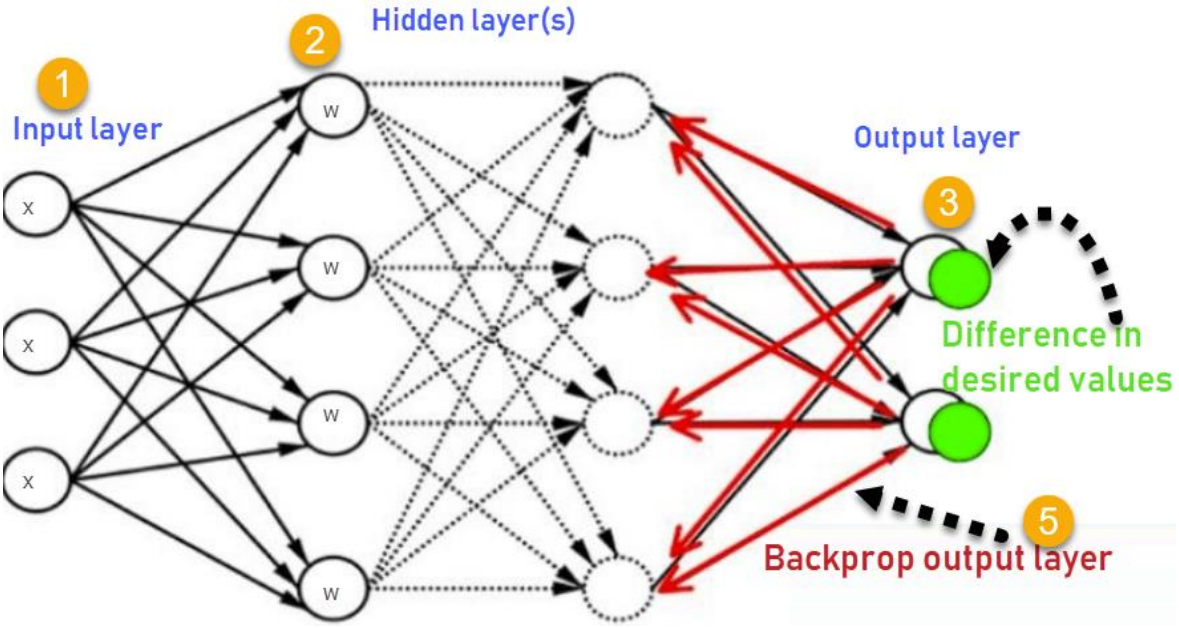
Multilayer Perceptron falls under the category of feedforward algorithms, because inputs are combined with the initial weights in a weighted sum and subjected to the activation function, just like in the Perceptron. But the difference is that each linear combination is propagated to the next layer.

Each layer is *feeding* the next one with the result of their computation, their internal representation of the data. This goes all the way through the hidden layers to the output layer.

Back-propagation algorithm and its variants Stochastic gradient decent

Backpropagation is the essence of neural network training. It is the method of fine-tuning the weights of a neural network based on the error rate obtained in the previous epoch (i.e., iteration). Proper tuning of the weights allows you to reduce error rates and make the model reliable by increasing its generalization.

Backpropagation in neural network is a short form for "backward propagation of errors." It is a standard method of training artificial neural networks. This method helps calculate the gradient of a loss function with respect to all the weights in the network.



Most prominent advantages of Backpropagation are:

- Backpropagation is fast, simple and easy to program
- It has no parameters to tune apart from the numbers of input
- It is a flexible method as it does not require prior knowledge about the network
- It is a standard method that generally works well
- It does not need any special mention of the features of the function to be learned.

What is a Feed Forward Network?

A feedforward neural network is an artificial neural network where the nodes never form a cycle. This kind of neural network has an input layer, hidden layers, and an output layer. It is the first and simplest type of artificial neural network.

Types of Backpropagation Networks

Two Types of Backpropagation Networks are:

- Static Back-propagation
- Recurrent Backpropagation

Static back-propagation:

It is one kind of backpropagation network which produces a mapping of a static input for static output. It is useful to solve static classification issues like optical character recognition.

Recurrent Backpropagation:

Recurrent Back propagation in data mining is fed forward until a fixed value is achieved. After that, the error is computed and propagated backward.

The main difference between both of these methods is: that the mapping is rapid in static back-propagation while it is non-static in recurrent backpropagation.

Gradient Descent

Gradient Descent is an iterative optimization process that searches for an objective function's optimum value (Minimum/Maximum). It is one of the most used methods for changing a model's parameters in order to reduce a cost function in machine learning projects.

The primary goal of gradient descent is to identify the model parameters that provide the maximum accuracy on both training and test datasets. In gradient descent, the gradient is a vector pointing in the general direction of the function's steepest rise at a particular point. The algorithm might gradually drop towards lower values of the function by moving in the opposite direction of the gradient, until reaching the minimum of the function.

Types of Gradient Descent:

Typically, there are three types of Gradient Descent:

1. Batch Gradient Descent
2. Stochastic Gradient Descent
3. Mini-batch Gradient Descent

Stochastic Gradient Descent Algorithm

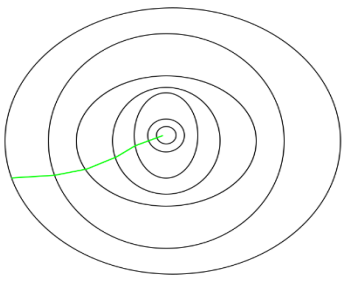
- **Initialization**: Randomly initialize the parameters of the model.
- **Set Parameters**: Determine the number of iterations and the learning rate (alpha) for updating the parameters.
- **Stochastic Gradient Descent Loop**: Repeat the following steps until the model converges or reaches the maximum number of iterations:
  a. Shuffle the training dataset to introduce randomness.

  b. Iterate over each training example (or a small batch) in the shuffled order.

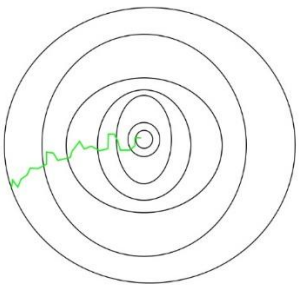
  c. Compute the gradient of the cost function with respect to the model parameters using the current training example (or batch).

  d. Update the model parameters by taking a step in the direction of the negative gradient, scaled by the learning rate.

  e. Evaluate the convergence criteria, such as the difference in the cost function between iterations of the gradient.



Batch gradient



Stochastic gradient descent

Curse of Dimensionality

Curse of Dimensionality describes the explosive nature of increasing data dimensions and its resulting exponential increase in computational efforts required for its processing and/or analysis. This term was first introduced by Richard E. Bellman, to explain the increase in volume of Euclidean space associated with adding extra dimensions, in area of dynamic programming. Today, this phenomenon is observed in fields like machine learning, data analysis, data mining to name a few. An increase in the dimensions can in theory, add more information to the data thereby improving the quality of data but practically increases the noise and redundancy during its analysis.

Behaviour of a Machine Learning Algorithms — Need for data points and Accuracy of Model

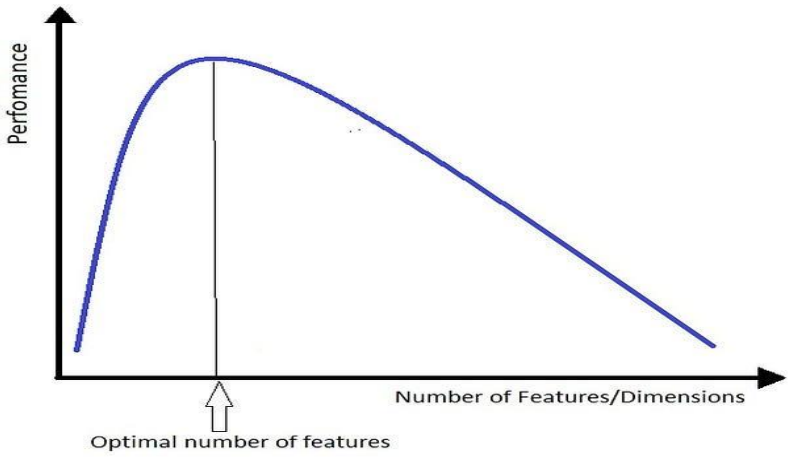
In machine learning, a feature of an object can be an attribute or a characteristic that defines it. Each feature represents a dimension and group of dimensions creates a data point. This represents a feature vector that defines the data point to be used by a machine learning algorithm(s). When we say increase in dimensionality it implies an increase in the number of features used to describe the data. For example, in the field of breast cancer research, age, number of cancerous nodes can be used as features to define the prognosis of the breast cancer patient. These features constitute the *dimensions* of a feature vector. But other factors like past surgeries, patient history, type of tumor and other such features help a doctor to better determine the prognosis. In this case by adding features, we are theoretically increasing the dimensions of our data.

As the dimensionality increases, the number of data points required for good performance of any machine learning algorithm increases exponentially. The reason is that, we would need more number of data points for any given combination of features, for any machine learning model to be valid. For example, let's say that for a model to perform well, we need at least 10 data points for each combination of feature values. If we assume that we have one binary feature, then for its 21 unique values (0 and 1) we would need 2^1x 10 = 20 data points. For 2 binary features, we would have 2^2 unique values and need 2^2 x 10 = 40 data points. Thus, for k-number of binary features we would need 2^k x 10 data points.

Need for Data Points with Increase in Dimensions

1 Binary feature →	2^1 unique values →	2^1 x 10 = 20 data points
2 Binary features →	2^2 unique values →	2^2 x 10 = 40 data points
3 Binary features →	2^3 unique values →	2^3 x 10 = 80 data points
.	.	.
.	.	.
.	.	.
k Binary features →	2^k unique values →	2^k x 10 data points

Hughes (1968) in his study concluded that with a fixed number of training samples, the predictive power of any classifier first increases as the number of dimensions increase, but after a certain value of number of dimensions, the performance deteriorates. Thus, the phenomenon of curse of dimensionality is also known as Hughes phenomenon.


Perfomance
Number of Features/Dimensions
Optimal number of features