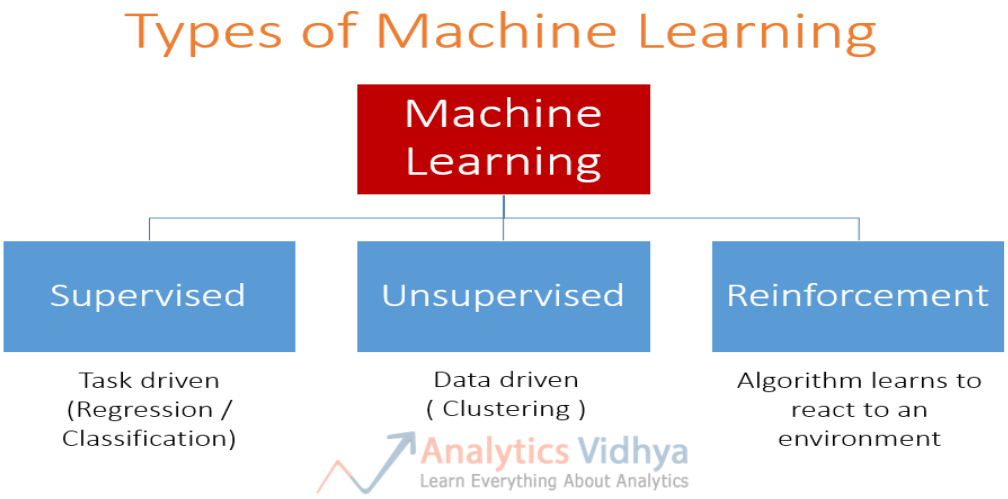
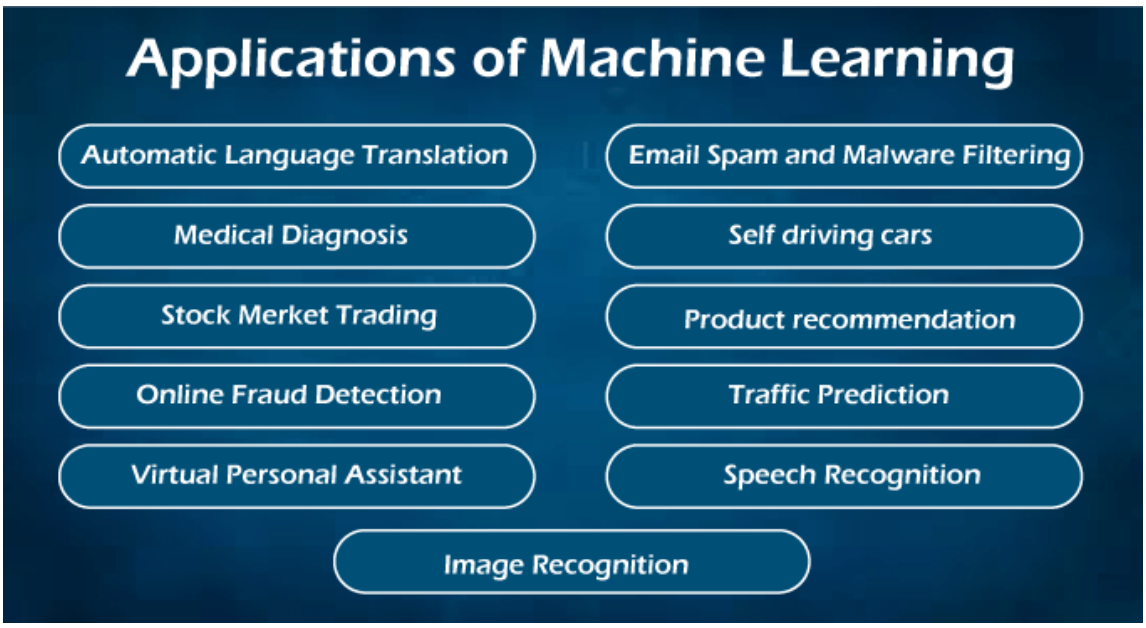
UNIT-2

TOPICS :Machine Learning: Basics and Under fitting, Hyper parameters and Validation Sets, Estimators, Bias and Variance, Maximum Likelihood, Bayesian Statistics,and Unsupervised Learning, Stochastic Gradient Descent, Challenges Motivating Deep Learning. Deep Feed forward Networks: Learning XOR, Gradient-Based Learning, Hidden Units, Architecture Design, Back-Propagation and other Differentiation Algorithms.

Machine Learning: Basics and Under fitting

Machine Learning is defined as a technology that is used to train machines to perform various actions such as predictions, recommendations, estimations, etc., based on historical data or past experience. Machine Learning enables computers to behave like human beings by training them with the help of past experience and predicted data.





Underfitting

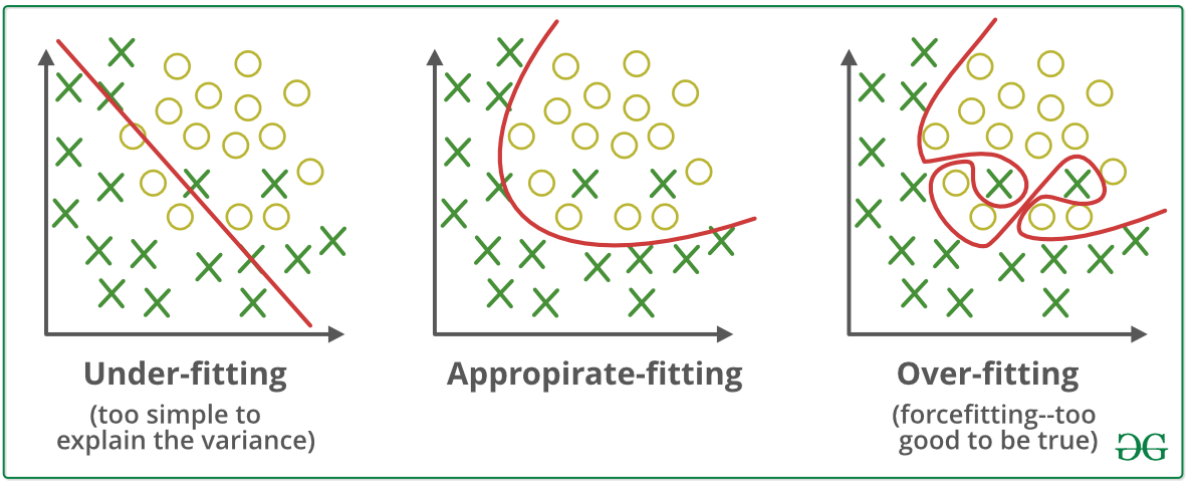
Underfitting: A statistical model or a machine learning algorithm is said to have underfitting when it cannot capture the underlying trend of the data, i.e., it only performs well on training data but performs poorly on testing data. Underfitting destroys the accuracy of our machine learning model. Its occurrence simply means that our model or the algorithm does not fit the data well enough. It usually happens when we have fewer data to build an accurate model and also when we try to build a linear model with fewer non-linear data. In such cases, the rules of the machine learning model are too easy and flexible to be applied to such minimal data and therefore the model will probably make a lot of wrong predictions. Underfitting can be avoided by using more data and also reducing the features by feature selection.
In a nutshell, Underfitting refers to a model that can neither performs well on the training data nor generalize to new data.

Reasons for Underfitting:

1. High bias and low variance
2. The size of the training dataset used is not enough.
3. The model is too simple.
4. Training data is not cleaned and also contains noise in it.

Techniques to reduce underfitting:

1. Increase model complexity
2. Increase the number of features, performing feature engineering
3. Remove noise from the data.
4. Increase the number of epochs or increase the duration of training to get better results.



Hyper parameters and Validation Sets

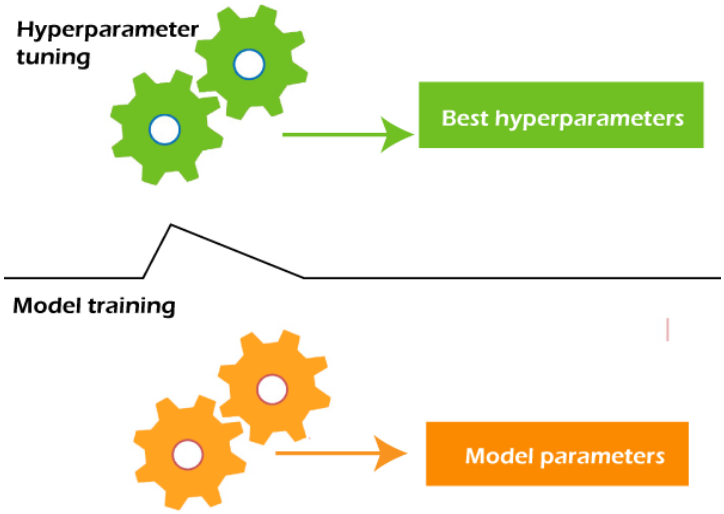
Hyper parameters:

A hyperparameter is a machine learning parameter whose value is chosen before a learning algorithm is trained. Hyperparameters should not be confused with parameters. In machine learning, the label *parameter* is used to identify variables whose values are learned during training.

Every variable that an AI engineer or ML engineer chooses before model training begins can be referred to as a hyperparameter -- as long as the value of the variable remains the same when training ends.

Examples of hyperparameters in machine learning include:

- Model architecture
- Learning rate
- Number of epochs
- Number of branches in a decision tree
- Number of clusters in a clustering algorithm



Train, Validation and Test Sets in Machine Learning

The Training Set:

It is the set of data that is used to train and make the model learn the hidden features/patterns in the data.

In each epoch, the same training data is fed to the neural network architecture repeatedly, and the model continues to learn the features of the data.

The training set should have a diversified set of inputs so that the model is trained in all scenarios and can predict any unseen data sample that may appear in the future.

Validation Sets:

The validation set is a set of data, separate from the training set, that is used to validate our model performance during training.

This validation process gives information that helps us tune the model's hyperparameters and configurations accordingly. It is like a critic telling us whether the training is moving in the right direction or not.

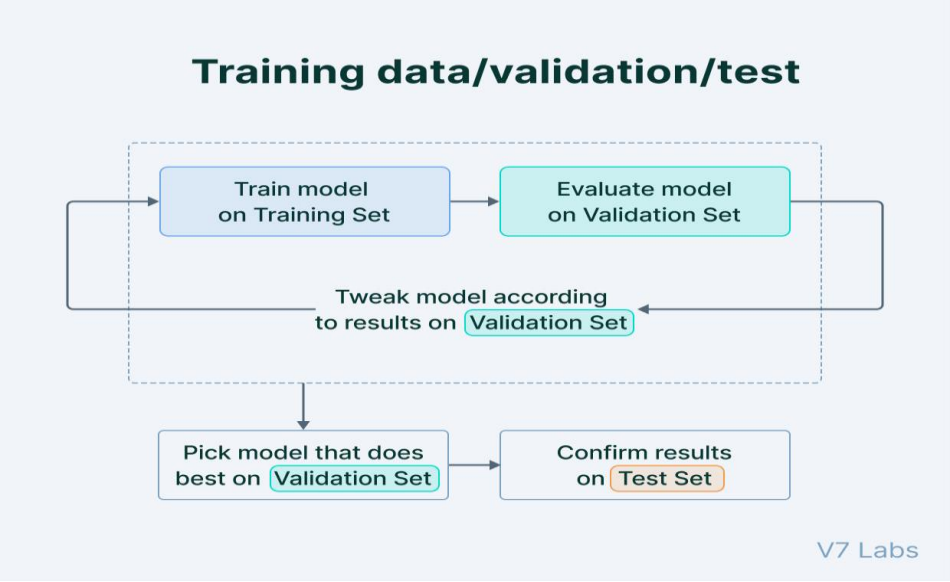
The model is trained on the training set, and, simultaneously, the model evaluation is performed on the validation set after every epoch.

The main idea of splitting the dataset into a validation set is to prevent our model from overfitting i.e., the model becomes really good at classifying the samples in the training set but cannot generalize and make accurate classifications on the data it has not seen before.

The Test Set:

The test set is a separate set of data used to test the model *after* completing the training.

It provides an unbiased final model performance metric in terms of accuracy, precision, etc. To put it simply, it answers the question of "*How well does the model perform?*"



Estimators

Estimators are functions of random variables that can help us find approximate values for these parameters. Think of these estimators like any other function, that takes an input, processes it, and renders an output. So, the process of estimation goes as follows:

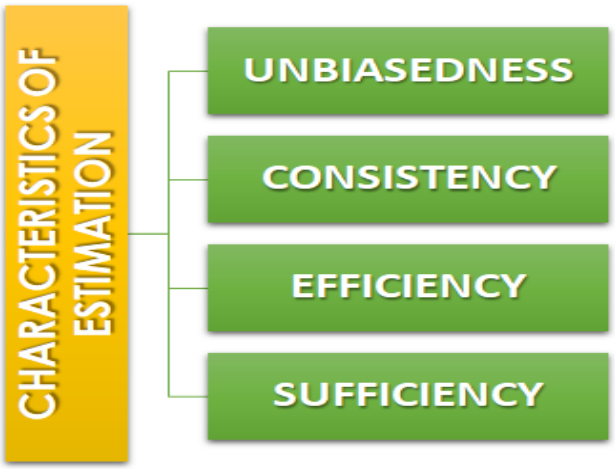
1) From the distribution, we take a series of random samples.

2) We input these random samples into the estimator function.

3) The estimator function processes it and gives a set of outputs.

4) The expected value of that set is the approximate value of the parameter.

- In machine learning, an estimator is **an equation for picking the "best," or most likely accurate, data model based upon observations in realty**. Not to be confused with estimation in general, the estimator is the formula that evaluates a given quantity (the estimand) and generates an estimate.
- **SVC, Naive Bayes, k-NN** are some of the popular classification estimators but have comparatively high time complexity. For data with <100k samples, one can try the Linear SVC model.
- An estimator is a statistic used for the purpose of **estimating an unknown parameter**. An estimator is a function of the data in a sample. Common estimators are the sample mean and sample variance which are used to estimate the unknown population mean and variance.
- The two main types of estimators in statistics are **point estimators and interval estimators**. Point estimation is the opposite of interval estimation. It produces a single value while the latter produces a range of values
- A good estimator should be **unbiased, consistent, and relatively efficient**.



In machine learning, an estimator is an equation for picking the "best," or most likely accurate, data model based upon observations in realty. Not to be confused with estimation in general, the estimator is the formula that evaluates a given quantity (the estimand) and generates an estimate. This estimate is then inserted into the deep learning classifier system to determine what action to take.

Uses of Estimators

By quantifying guesses, estimators are how machine learning in theory is implemented in practice. Without the ability to estimate the parameters of a dataset (such as the layers in a neural network or the bandwidth in a kernel), there would be no way for an AI system to "learn."

A simple example of estimators and estimation in practice is the so-called "German Tank Problem" from World War Two. The Allies had no way to know for sure how many tanks the Germans were building every month. By counting the serial numbers of captured or destroyed tanks (the estimand), Allied statisticians created an estimator rule. This equation calculated the maximum possible number of tanks based upon the sequential serial numbers, and apply minimum variance analysis to generate the most likely estimate for how many new tanks German was building.

Types of Estimators

Estimators come in two broad categories—point and interval. Point equations generate single value results, such as standard deviation, that can be plugged into a deep learning algorithm's classifier functions. Interval equations generate a range of likely values, such as a confidence interval, for analysis.

In addition, each estimator rule can be tailored to generate different types of estimates:

- Biased - Either an overestimate or an underestimate.
- Efficient - Smallest variance analysis. The smallest possible variance is referred to as the "best" estimate.
- Invariant: Less flexible estimates that aren't easily changed by data transformations.
- Shrinkage: An unprocessed estimate that's combined with other variables to create complex estimates.
- Sufficient: Estimating the total population's parameter from a limited dataset.
- Unbiased: An exact-match estimate value that neither underestimates nor overestimates.

Bias and Variance

What is bias in machine learning?

Bias is a phenomenon that skews the result of an algorithm in favor or against an idea.

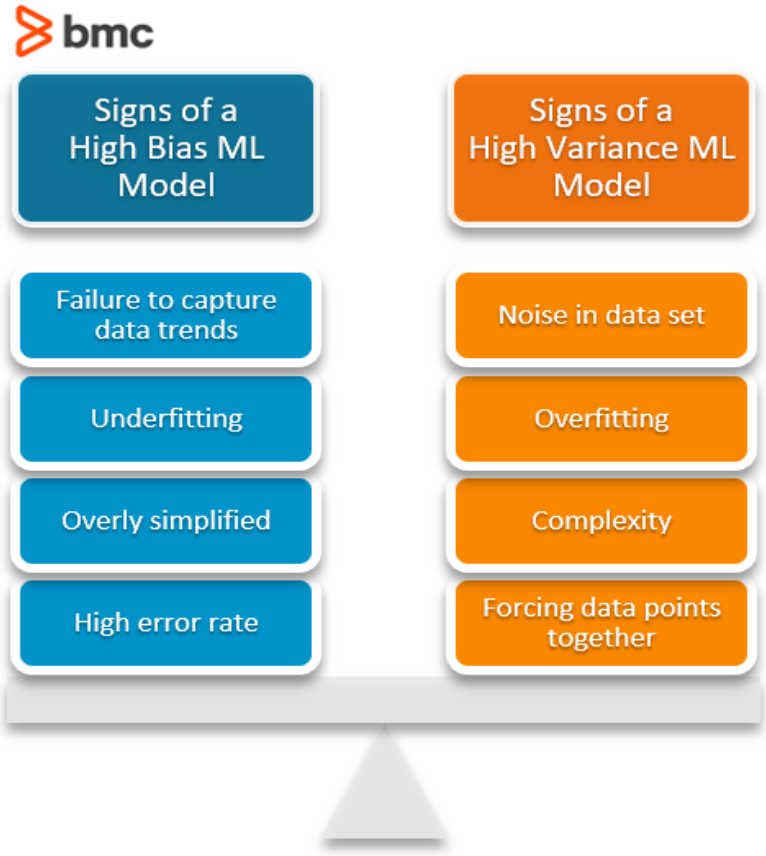
Bias is considered a systematic error that occurs in the machine learning model itself due to incorrect assumptions in the ML process.

Technically, we can define bias as the error between average model prediction and the ground truth. Moreover, it describes how well the model matches the training data set:

- A model with a higher bias would not match the data set closely.
- A low bias model will closely match the training data set.

Characteristics of a high bias model include:

- Failure to capture proper data trends
- Potential towards underfitting
- More generalized/overly simplified
- High error rate



What is variance in machine learning?

Variance refers to the changes in the model when using different portions of the training data set.

Simply stated, variance is the variability in the model prediction—how much the ML function can adjust depending on the given data set. Variance comes from highly complex models with a large number of features.

- Models with high bias will have low variance.
- Models with high variance will have a low bias.

All these contribute to the flexibility of the model. For instance, a model that does not match a data set with a high bias will create an inflexible model with a low variance that results in a suboptimal machine learning model.

Characteristics of a high variance model include:

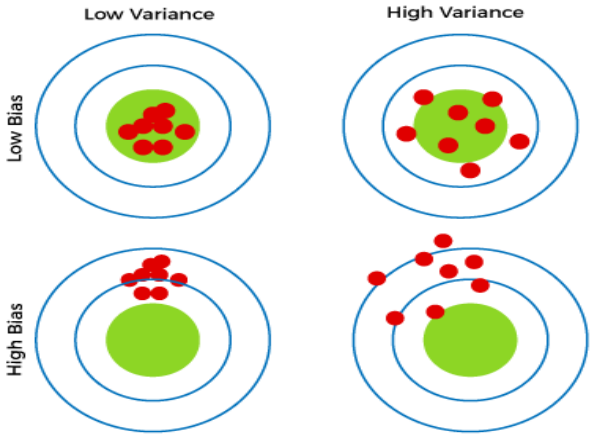
- Noise in the data set
- Potential towards overfitting
- Complex models
- Trying to put all data points as close as possible

This table lists common algorithms and their expected behavior regarding bias and variance:

Algorithm	**Bias**	**Variance**
Linear Regression	High Bias	Less Variance
Decision Tree	Low Bias	High Variance
Bagging	Low Bias	High Variance (Less than Decision Tree)
Random Forest	Low Bias	High Variance (Less than Decision Tree and Bagging)

Different Combinations of Bias-Variance

There are four possible combinations of bias and variances, which are represented by the below diagram:

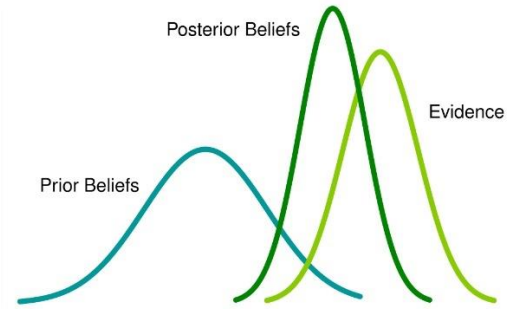


1. **Low-Bias,Low-Variance:**
   The combination of low bias and low variance shows an ideal machine learning model. However, it is not possible practically.
2. **Low-Bias, High-Variance:** With low bias and high variance, model predictions are inconsistent and accurate on average. This case occurs when the model learns with a large number of parameters and hence leads to an **overfitting**
3. **High-Bias, Low-Variance:** With High bias and low variance, predictions are consistent but inaccurate on average. This case occurs when a model does not learn well with the training dataset or uses few numbers of the parameter. It leads to **underfitting** problems in the model.
4. **High-Bias,High-Variance:**
   With high bias and high variance, predictions are inconsistent and also inaccurate on average.

Bayesian Statistics in Machine Learning

- Bayesian Statistics are a technique that assigns “degrees of belief,” or Bayesian probabilities, to traditional statistical modeling. In this interpretation of statistics, probability is calculated as the reasonable expectation of an event occurring based upon currently known triggers. Or in other words, that probability is a dynamic process that can change as new information is gathered, rather than a fixed value based upon frequency or propensity.
- Bayesian statistics is **an approach to data analysis and parameter estimation based on Bayes' theorem**. Unique for Bayesian statistics is that all observed and unobserved parameters in a statistical model are given a joint probability distribution, termed the prior and data distributions.
- Bayesian inference is **a specific way to learn from data that is heavily used in statistics for data analysis**. Bayesian inference is used less often in the field of machine learning, but it offers an elegant framework to understand what “learning” actually is. It is generally useful to know about Bayesian inference.

- **Bayesian inference uses Bayesian probability to summarize evidence for the likelihood of a prediction**. - Bayesian statistics helps some models by classifying and specifying the prior distributions of any unknown parameters.
- Bayesian statistics is a particular approach to **applying probability to statistical problems**. It provides us with mathematical tools to update our beliefs about random events in light of seeing new data or evidence about those events.
- Bayesian strategies help various machine learning algorithms in removing pivotal data from little informational indexes and taking care of missing information. They assume a significant function in an immense scope of territories from game improvement to sedate disclosure.



Thus, this statistical approach is not directly applicable to every deep learning technique. However, it affects the three key fields of machine learning:

1. **Statistical Inference- It** uses Bayesian probability, to sum up proof for the likelihood of an expectation.
2. **Statistical Modeling-** It encourages a few models by grouping and indicating the earlier dissemination of any obscure boundaries.
3. **Experiment Design-** By including the idea of "prior belief influence," this strategy utilizes successive investigations to factor in the result of prior tests when planning new ones. These "beliefs" are refreshed by prior and posterior dispersion.

Uses of Bayesian statistics

While most machine learning algorithms attempt to foresee results from enormous datasets, the Bayesian methodology is useful for a few classes of issues that aren't effectively fathomed with other likelihood models. Specifically;

- Information bases with not many information focus for reference,
- Models with solid earlier instincts from previous perceptions,
- Information with elevated levels of uncertainty, or when it's important to evaluate the degree of uncertainty over a whole model or look at changed models.

At the point when a model creates an invalid theory yet, it's important to guarantee something about the probability of the elective speculation.

EXAMPLE:

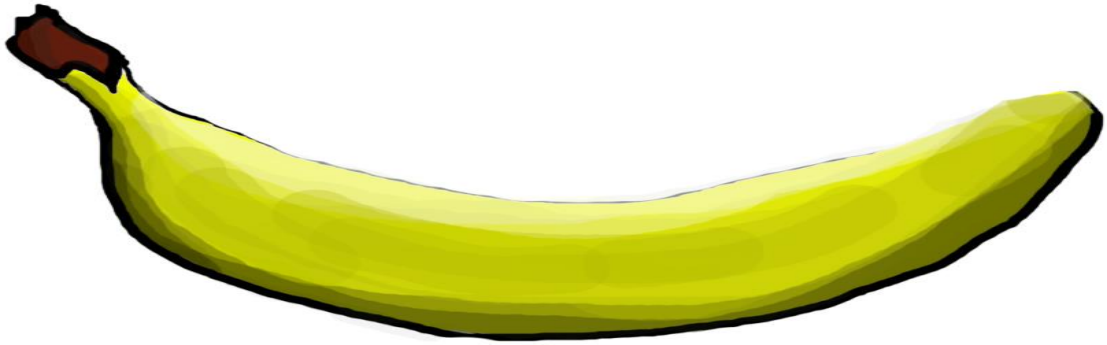
Weather prediction: Bayesian inference can be used in Bayesian machine learning to predict the weather with more accuracy. Bayes' theorem can be applied for predicting real-time weather patterns and probabilities of rain based on past data such as temperature, humidity, etc.

Supervised learning: Supervised learning, as the name indicates, has the presence of a supervisor as a teacher. Basically supervised learning is when we teach or train the machine using data that is well-labelled. Which means some data is already tagged with the correct answer. After that, the machine is provided with a new set of examples(data) so that the supervised learning algorithm analyses the training data(set of training examples) and produces a correct outcome from labeled data.
For instance, suppose you are given a basket filled with different kinds of fruits. Now the first step is to train the machine with all the different fruits one by one like this:

If the shape of the object is rounded and has a depression at the top, is red in color, then it will be labeled as –Apple.

If the shape of the object is a long curving cylinder having Green-Yellow color, then it will be labeled as –Banana.

Now suppose after training the data, you have given a new separate fruit, say Banana from the basket, and asked to identify it.



Since the machine has already learned the things from previous data and this time has to use it wisely. It will first classify the fruit with its shape and color and would confirm the fruit name as BANANA and put it in the Banana category. Thus the machine learns the things from training data(basket containing fruits) and then applies the knowledge to test data(new fruit).

Supervised learning is classified into two categories of algorithms:

- **Classification**: A classification problem is when the output variable is a category, such as “Red” or “blue” , “disease” or “no disease”.
- **Regression**: A regression problem is when the output variable is a real value, such as “dollars” or “weight”.

Supervised learning deals with or learns with “labeled” data. This implies that some data is already tagged with the correct answer.

Types:-

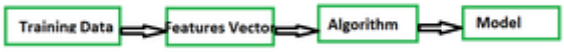
- Regression
- Logistic Regression
- Classification
- Naive Bayes Classifiers
- K-NN (k nearest neighbors)
- Decision Trees
- Support Vector Machine

Advantages:-

- Supervised learning allows collecting data and produces data output from previous experiences.
- Helps to optimize performance criteria with the help of experience.
- Supervised machine learning helps to solve various types of real-world computation problems.
- It performs classification and regression tasks.
- It allows estimating or mapping the result to a new sample.
- We have complete control over choosing the number of classes we want in the training data.

Disadvantages:-

- Classifying big data can be challenging.
- Training for supervised learning needs a lot of computation time. So, it requires a lot of time.
- Supervised learning cannot handle all complex tasks in Machine Learning.
- Computation time is vast for supervised learning.
- It requires a labelled data set.
- It requires a training process.



Steps

Unsupervised Learning

- *Unsupervised learning is a type of machine learning in which models are trained using unlabeled dataset and are allowed to act on that data without any supervision.*
- Unsupervised learning cannot be directly applied to a regression or classification problem because unlike supervised learning, we have the input data but no corresponding output data. The goal of unsupervised learning is to **find the underlying structure of dataset, group that data according to similarities, and represent that dataset in a compressed format**.

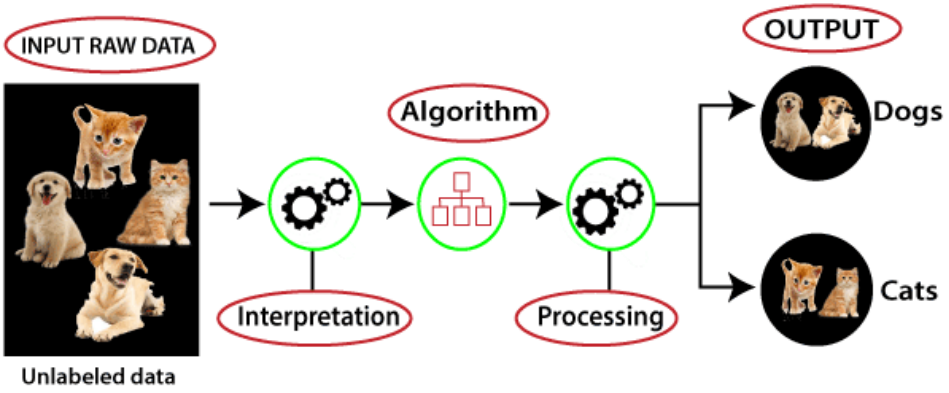
Why use Unsupervised Learning?

Below are some main reasons which describe the importance of Unsupervised Learning:

- Unsupervised learning is helpful for finding useful insights from the data.
- Unsupervised learning is much similar as a human learns to think by their own experiences, which makes it closer to the real AI.
- Unsupervised learning works on unlabeled and uncategorized data which make unsupervised learning more important.
- In real-world, we do not always have input data with the corresponding output so to solve such cases, we need unsupervised learning.

Working of Unsupervised Learning

Working of unsupervised learning can be understood by the below diagram:

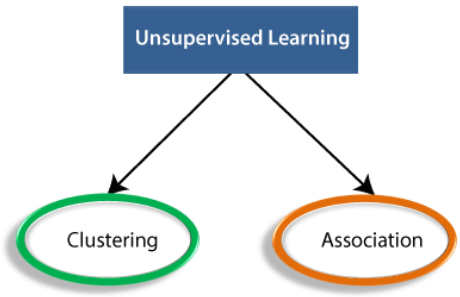


Here, we have taken an unlabeled input data, which means it is not categorized and corresponding outputs are also not given. Now, this unlabeled input data is fed to the machine learning model in order to train it. Firstly, it will interpret the raw data to find the hidden patterns from the data and then will apply suitable algorithms such as k-means clustering, Decision tree, etc.

Once it applies the suitable algorithm, the algorithm divides the data objects into groups according to the similarities and difference between the objects.

Types of Unsupervised Learning Algorithm:

The unsupervised learning algorithm can be further categorized into two types of problems:



- **Clustering**: Clustering is a method of grouping the objects into clusters such that objects with most similarities remains into a group and has less or no similarities with the objects of another group. Cluster analysis finds the commonalities between the data objects and categorizes them as per the presence and absence of those commonalities.
- **Association**: An association rule is an unsupervised learning method which is used for finding the relationships between variables in the large database. It determines the set of items that occurs together in the dataset. Association rule makes marketing strategy more effective. Such as people who buy X item (suppose a bread) are also tend to

purchase Y (Butter/Jam) item. A typical example of Association rule is Market Basket Analysis.

Unsupervised Learning algorithms:

Below is the list of some popular unsupervised learning algorithms:

- **K-means clustering**
- **KNN (k-nearest neighbors)**
- **Hierarchal clustering**
- **Anomaly detection**
- **Neural Networks**
- **Principle Component Analysis**
- **Independent Component Analysis**
- **Apriori algorithm**
- **Singular value decomposition**

Advantages of Unsupervised Learning

- Unsupervised learning is used for more complex tasks as compared to supervised learning because, in unsupervised learning, we don't have labelled input data.
- Unsupervised learning is preferable as it is easy to get unlabeled data in comparison to labelled data.

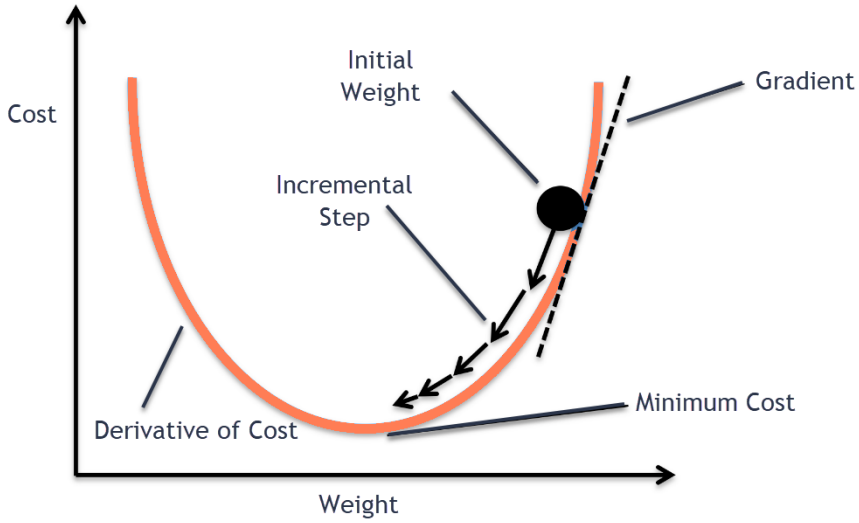
Disadvantages of Unsupervised Learning

- Unsupervised learning is intrinsically more difficult than supervised learning as it does not have corresponding output.
- The result of the unsupervised learning algorithm might be less accurate as input data is not labelled, and algorithms do not know the exact output in advance.

Stochastic Gradient Descent

- Stochastic gradient descent is **an optimization algorithm often used in machine learning applications to find the model parameters that correspond to the best fit between predicted and actual outputs**. It's an inexact but powerful technique. Stochastic gradient descent is widely used in machine learning applications.

- Stochastic gradient descent is a popular algorithm for training a wide range of models in machine learning, including **(linear) support vector machines, logistic regression (see, e.g., Vowpal Wabbit) and graphical models**.



Challenges Motivating Deep Learning

In Machine Learning, there occurs a process of analyzing data for building or training models. It is just everywhere; from Amazon product recommendations to self-driven cars, it beholds great value throughout. As per the latest research, the global machine learning market is expected to grow by 43% by 2024. This revolution has enhanced the demand for machine learning professionals to a great extent. AI and machine learning jobs have observed a significant growth rate of 75% in the past four years, and the industry is growing continuously. A career in the Machine learning domain offers job satisfaction, excellent growth, insanely high salary, but it is a complex and challenging process.

There are a lot of challenges that machine learning professionals face to inculcate ML skills and create an application from scratch. What are these challenges? In this blog, we will discuss seven major challenges faced by machine learning professionals. Let’s have a look.

1. Poor Quality of Data

Data plays a significant role in the machine learning process. One of the significant issues that machine learning professionals face is the absence of good quality data. Unclean and noisy data can make the whole process extremely exhausting. We don’t want our algorithm to make inaccurate or faulty predictions.

Hence the quality of data is essential to enhance the output. Therefore, we need to ensure that the process of data preprocessing which includes removing outliers, filtering missing values, and removing unwanted features, is done with the utmost level of perfection.

2. Underfitting of Training Data

This process occurs when data is unable to establish an accurate relationship between input and output variables. It simply means trying to fit in undersized jeans. It signifies the data is too simple to establish a precise relationship. To overcome this issue:

- *Maximize the training time*
- *Enhance the complexity of the model*
- *Add more features to the data*
- *Reduce regular parameters*
- *Increasing the training time of model*

3. Overfitting of Training Data

Overfitting refers to a machine learning model trained with a massive amount of data that negatively affect its performance. It is like trying to fit in Oversized jeans. Unfortunately, this is one of the significant issues faced by machine learning professionals. This means that the algorithm is trained with noisy and biased data, which will affect its overall performance. Let's understand this with the help of an example. Let's consider a model trained to differentiate between a cat, a rabbit, a dog, and a tiger. The training data contains 1000 cats, 1000 dogs, 1000 tigers, and 4000 Rabbits. Then there is a considerable probability that it will identify the cat as a rabbit. In this example, we had a vast amount of data, but it was biased; hence the prediction was negatively affected.

We can tackle this issue by:

- *Analyzing the data with the utmost level of perfection*
- *Use data augmentation technique*
- *Remove outliers in the training set*
- *Select a model with lesser features*

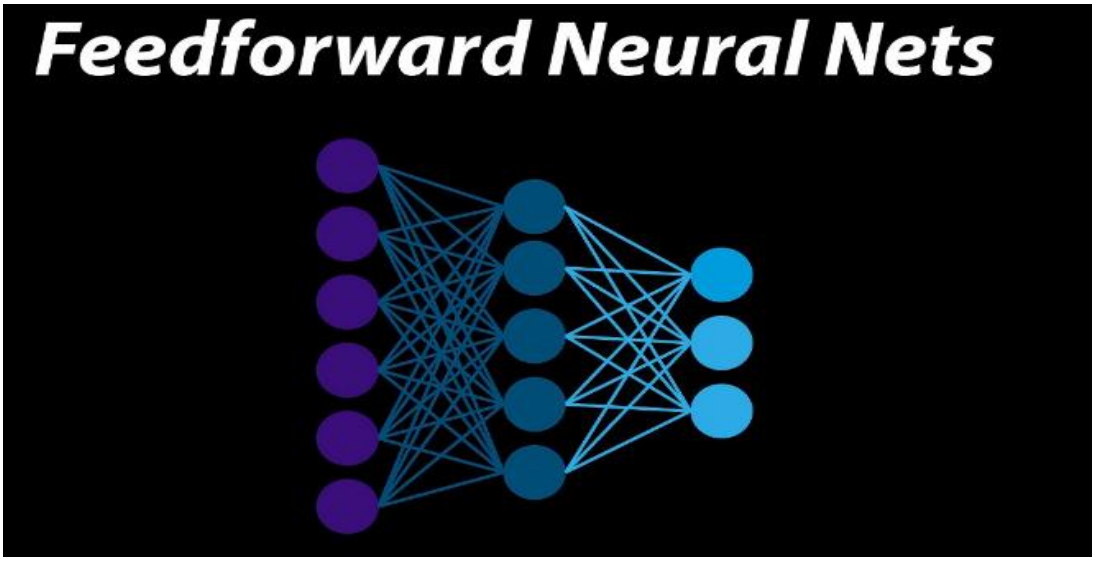
- **4. Machine Learning is a Complex Process**
- The machine learning industry is young and is continuously changing. Rapid hit and trial experiments are being carried on. The process is transforming, and hence there are high chances of error which makes the learning complex. It includes analyzing the data, removing data bias, training data, applying complex mathematical calculations, and a lot more. Hence it is a really complicated process which is another big challenge for Machine learning professionals.
- **5. Lack of Training Data**
  The most important task you need to do in the machine learning process is to train the data to achieve an accurate output. Less amount training data will produce inaccurate or too biased predictions. Let us understand this with the help of an example. Consider a machine learning algorithm similar to training a child. One day you decided to explain to a child how to distinguish between an apple and a watermelon. You will take an apple and a watermelon and show him the difference

between both based on their color, shape, and taste. In this way, soon, he will attain perfection in differentiating between the two. But on the other hand, a machine-learning algorithm needs a lot of data to distinguish. For complex problems, it may even require millions of data to be trained. Therefore we need to ensure that Machine learning algorithms are trained with sufficient amounts of data.

- **6. Slow Implementation**
  This is one of the common issues faced by machine learning professionals. The machine learning models are highly efficient in providing accurate results, but it takes a tremendous amount of time. Slow programs, data overload, and excessive requirements usually take a lot of time to provide accurate results. Further, it requires constant monitoring and maintenance to deliver the best output.
- **7. Imperfections in the Algorithm When Data Grows**
  So you have found quality data, trained it amazingly, and the predictions are really concise and accurate. Yay, you have learned how to create a machine learning algorithm!! But wait, there is a twist; the model may become useless in the future as data grows. The best model of the present may become inaccurate in the coming Future and require further rearrangement. So you need regular monitoring and maintenance to keep the algorithm working. This is one of the most exhausting issues faced by machine learning professionals.
  Conclusion: Machine learning is all set to bring a big bang transformation in technology. It is one of the most rapidly growing technologies used in medical diagnosis, speech recognition, robotic training, product recommendations, video surveillance, and this list goes on. This continuously evolving domain offers immense job satisfaction, excellent opportunities, global exposure, and exorbitant salary. It is a high risk and a high return technology. Before starting your machine learning journey, ensure that you carefully examine the challenges mentioned above. To learn this fantastic technology, you need to plan carefully, stay patient, and maximize your efforts.

Deep Feed forward Networks:

A Feed Forward Neural Network is an artificial Neural Network in which the nodes are connected circularly. A feed-forward neural network, in which some routes are cycled, is the polar opposite of a Recurrent Neural Network. The feed-forward model is the basic type of neural network because the input is only processed in one direction. The data always flows in one direction and never backwards/opposite.



Why are Neural Networks used?

Neural Networks are a type of function that connects inputs with outputs. In theory, neural networks should be able to estimate any sort of function, no matter how complex it is. Nonetheless, supervised learning entails learning a function that translates a given X to a specified Y and then utilising that function to determine the proper Y for a fresh X. If that's the case, how do neural networks differ from typical machine learning methods? Inductive Bias, a psychological phenomenon, is the answer. The phrase may appear to be fresh. However, before applying a machine learning model to it, it is nothing more than our assumptions about the relationship between X and Y.

The linear relationship between X and Y is the Inductive Bias of linear regression. As a result, it fits the data to a line or a hyperplane.

When there is a non-linear and complex relationship between X and Y, nevertheless, a Linear Regression method may struggle to predict Y. To approximate that relationship, we may need a curve or a multi-dimensional curve in this scenario.

A Feed Forward Neural Network is an artificial Neural Network in which the nodes are connected circularly. A feed-forward neural network, in which some routes are cycled, is the polar opposite of a Recurrent Neural Network. The feed-forward model is the basic type of

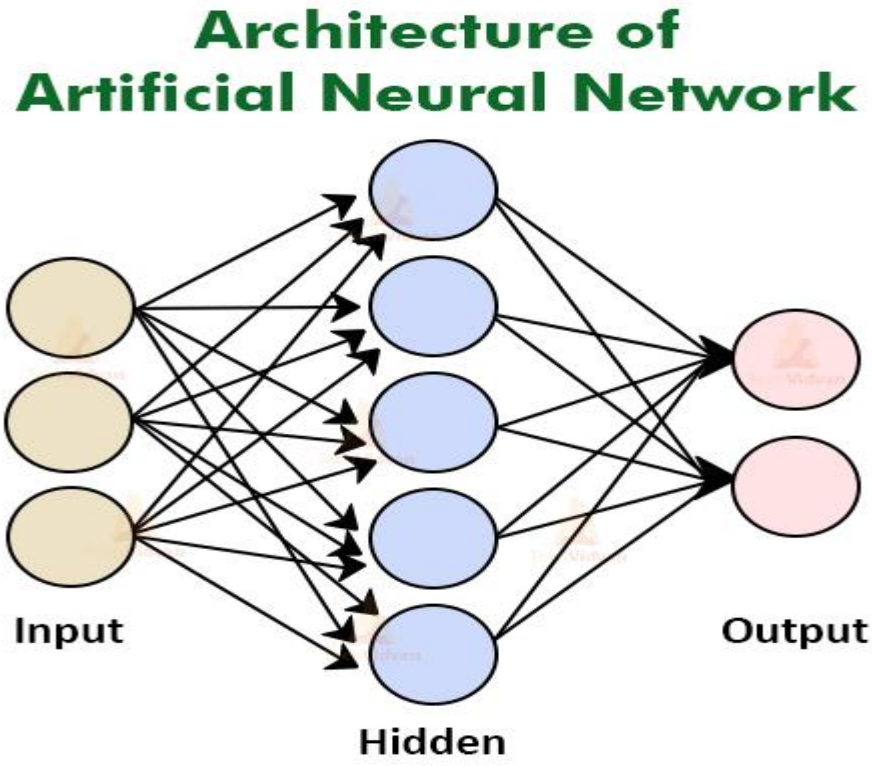
neural network because the input is only processed in one direction. The data always flows in one direction and never backwards/opposite.

The Neural Network advanced from the perceptron, a prominent machine learning algorithm. Frank Rosenblatt, a physicist, invented perceptrons in the 1950s and 1960s, based on earlier work by Warren McCulloch and Walter Pitts.

Neural Network Architecture and Operation

Before we look at why neural networks work, it's important to understand what neural networks do. Before we can grasp the design of a neural network, we must first understand what a neuron performs.

A weight is assigned to each input to an artificial neuron. First, the inputs are multiplied by their weights, and then a bias is applied to the outcome. After that, the weighted sum is passed via an activation function, being a non-linear function.



weight is being applied to each input to an artificial neuron. First, the inputs are multiplied by their weights, and then a bias is applied to the outcome. This is called

the weighted sum. After that, the weighted sum is processed via an activation function, as a non-linear function.

The first layer is the input layer, which appears to have six neurons but is only the data that is sent into the neural network. The output layer is the final layer. The dataset and the type of challenge determine the number of neurons in the final layer and the first layer. Trial and error will be used to determine the number of neurons in the hidden layers and the number of hidden layers.

All of the inputs from the previous layer will be connected to the first neuron from the first hidden layer. The second neuron in the first hidden layer will be connected to all of the preceding layer's inputs, and so forth for all of the first hidden layer's neurons. The outputs of the previously hidden layer are regarded inputs for neurons in the second hidden layer, and each of these neurons is coupled to all of the preceding neurons.

This article was published as a part of the Data Science Blogathon.

A Feed Forward Neural Network is an artificial Neural Network in which the nodes are connected circularly. A feed-forward neural network, in which some routes are cycled, is the polar opposite of a Recurrent Neural Network. The feed-forward model is the basic type of neural network because the input is only processed in one direction. The data always flows in one direction and never backwards/opposite.

Why are Neural Networks used?
Neural Networks are a type of function that connects inputs with outputs. In theory, neural networks should be able to estimate any sort of function, no matter how complex it is. Nonetheless, supervised learning entails learning a function that translates a given X to a specified Y and then utilising that function to determine the proper Y for a fresh X. If that's the case, how do neural networks differ from typical machine learning methods? Inductive Bias, a psychological phenomenon, is the answer. The phrase may appear to be fresh. However, before applying a machine learning model to it, it is nothing more than our assumptions about the relationship between X and Y.

The linear relationship between X and Y is the Inductive Bias of linear regression. As a result, it fits the data to a line or a hyperplane.

When there is a non-linear and complex relationship between X and Y, nevertheless, a Linear Regression method may struggle to predict Y. To approximate that relationship, we may need a curve or a multi-dimensional curve in this scenario.

However, depending on the function's complexity, we may need to manually set the number of neurons in each layer and the total number of layers in the network. This is usually accomplished through trial and error methods as well as experience. As a result, these parameters are referred to as hyperparameters.

Neural Network Architecture and Operation

Before we look at why neural networks work, it's important to understand what neural networks do. Before we can grasp the design of a neural network, we must first understand what a neuron performs.

A weight is assigned to each input to an artificial neuron. First, the inputs are multiplied by their weights, and then a bias is applied to the outcome. After that, the weighted sum is passed via an activation function, being a non-linear function.

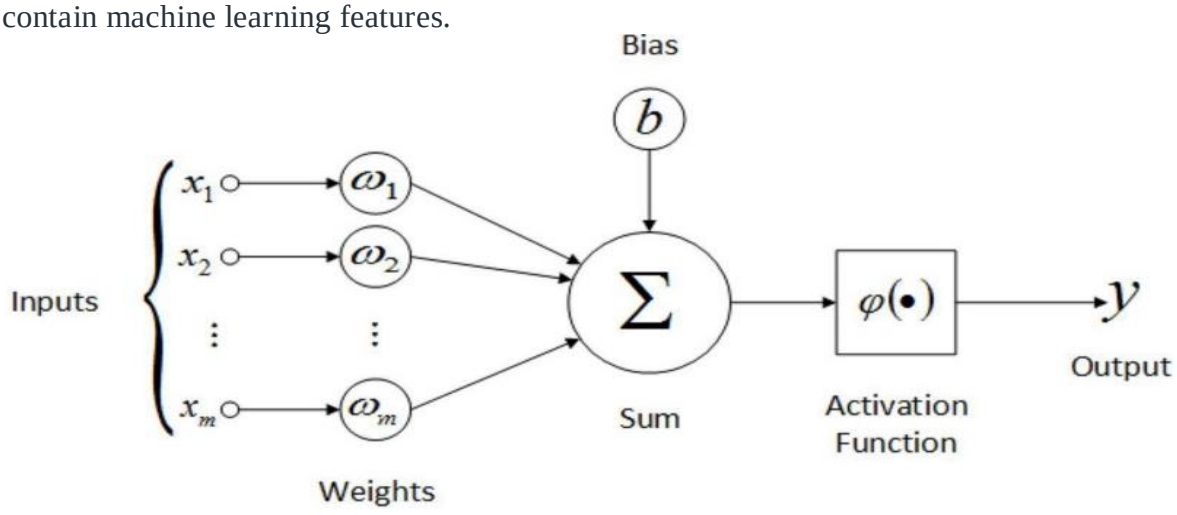
Artificial Neural Network

A weight is being applied to each input to an artificial neuron. First, the inputs are multiplied by their weights, and then a bias is applied to the outcome. This is called the weighted sum. After that, the weighted sum is processed via an activation function, as a non-linear function.

The first layer is the input layer, which appears to have six neurons but is only the data that is sent into the neural network. The output layer is the final layer. The dataset and the type of challenge determine the number of neurons in the final layer and the first layer. Trial and error will be used to determine the number of neurons in the hidden layers and the number of hidden layers.

All of the inputs from the previous layer will be connected to the first neuron from the first hidden layer. The second neuron in the first hidden layer will be connected to all of the preceding layer's inputs, and so forth for all of the first hidden layer's neurons. The outputs of the previously hidden layer are regarded inputs for neurons in the second hidden layer, and each of these neurons is coupled to all of the preceding neurons.

What is a Feed-Forward Neural Network and how does it work?
In its most basic form, a Feed-Forward Neural Network is a single layer perceptron. A sequence of inputs enter the layer and are multiplied by the weights in this model. The weighted input values are then summed together to form a total. If the sum of the values is more than a predetermined threshold, which is normally set at zero, the output value is usually 1, and if the sum is less than the threshold, the output value is usually -1. The single-layer perceptron is a popular feed-forward neural network model that is frequently used for classification. Single-layer perceptrons can also contain machine learning features.



The neural network can compare the outputs of its nodes with the desired values using a property known as the delta rule, allowing the network to alter its weights through training to create more accurate output values. This training and learning procedure results in gradient descent. The technique of updating weights in multi-layered perceptrons is virtually the same, however, the process is referred to as back-propagation. In such circumstances, the output values provided by the final layer are used to alter each hidden layer inside the network.

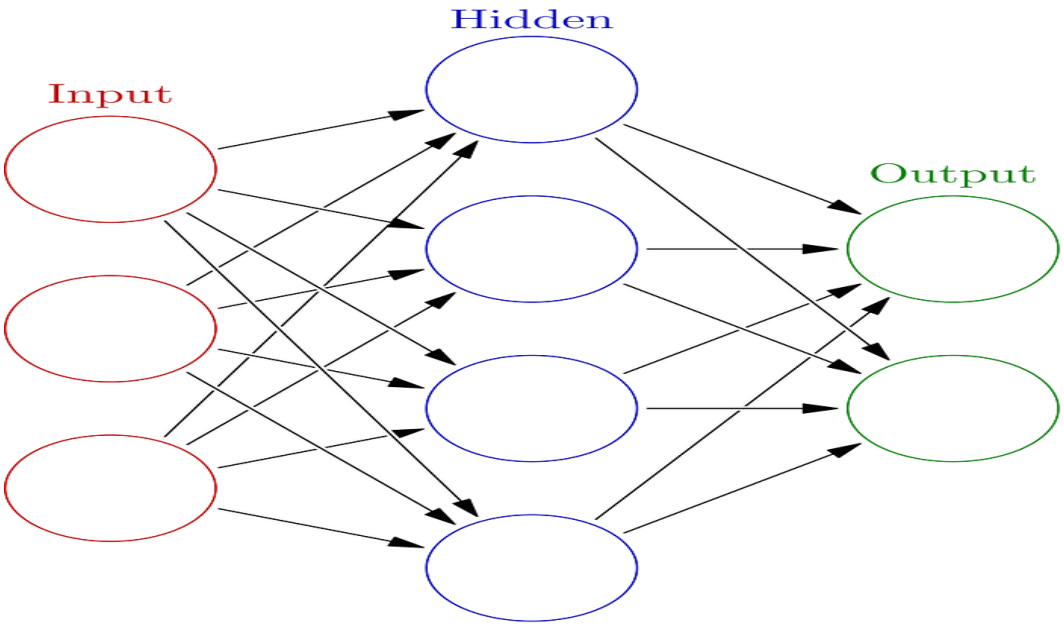
Deep Feedforward Networks:Learning XOR:

Deep feedforward networks, also often called feedforward neural networks, or multilayer perceptrons (MLPs), are the quint essential deep learning models.

These models are called feed forward because information flows through the function being evaluated from x, through the intermediate computations used to define f, and finally to the output y. There are no feedback connections in which outputs of the model are fed back into itself. When feedforward neural networks are extended to include feedback connections, they are called recurrent neural networks,

Feedforward networks are of extreme importance to machine learning practitioners. They form the basis of many important commercial applications. For example, the convolutional networks used for object recognition from photos are a specialized kind of feedforward network. Feedforward networks are a conceptual stepping stone on the path to recurrent networks, which power many natural language applications.

Feedforward neural networks are called networks because they are typically represented by composing together many different functions. The model is associated with a directed acyclic graph describing how the functions are composed together. For example, we might have three functions f(1), f(2), and f(3) connected in a chain, to form f(x) = f(3)(f(2)(f(1)(x))). These chain structures are the most commonly used structures of neural networks. In this case, f (1) is called the first layer of the network, f(2) is called the second layer, and so on.



Feed Forward network

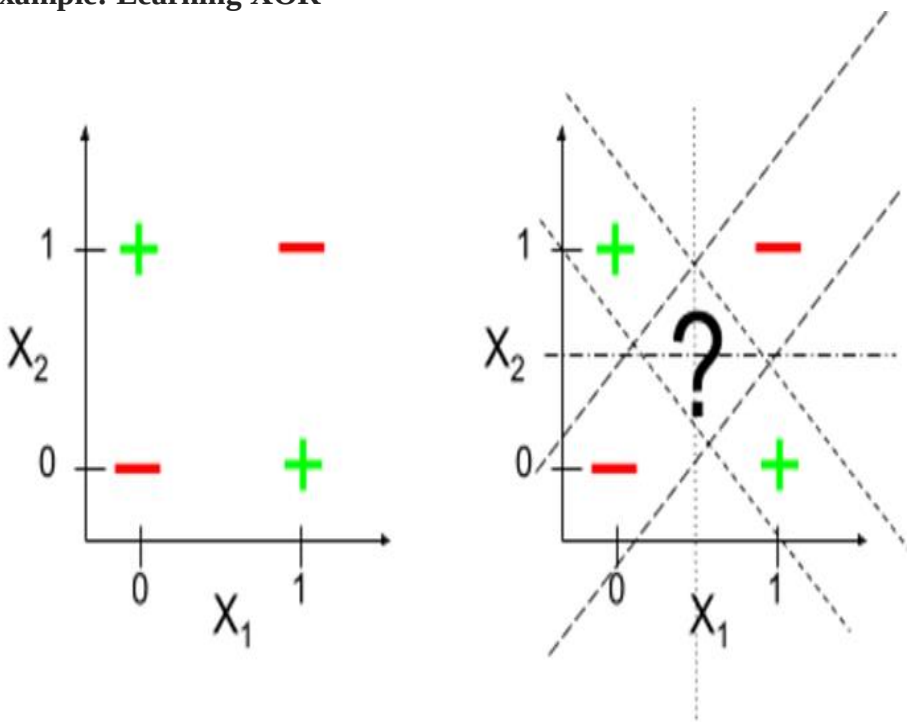
The goal of a feedforward network is to approximate some function f . For example, for a classifier, $y = f*(x)$ maps an input x to a category y. A feedforward network defines a mapping $y = f(x; \theta)$ and learns the value of the parameters θ that result in the best function approximation. length of the chain gives the depth of the model. It is from this terminology that the name "deep learning" arises. The final layer of a feedforward network is called the output layer. During neural network training, we drive f(x) to match $f*(x)$.

The training data provides us with noisy, approximate examples of $f*(x)$ evaluated at different training points. Each example x is accompanied by a label $y \approx f*(x)$.

The training examples specify directly what the output layer must do at each point x; it must produce a value that is close to y. The behavior of the other layers is not directly specified by the training data. The learning algorithm must decide how to use those layers to produce the desired output, but the training data does not say what each individual layer should do.

Finally, these networks are called neural because they are loosely inspired by neuroscience. Each hidden layer of the network is typically vector-valued. The dimensionality of these hidden layers determines the width of the model. Each element of the vector may be interpreted as playing a role analogous to a neuron. Rather than thinking of the layer as representing a single vector-to-vector function,

Example: Learning XOR



To make the idea of a feedforward network more concrete, we begin with an example of a fully functioning feedforward network on a very simple task: learning the XOR function.

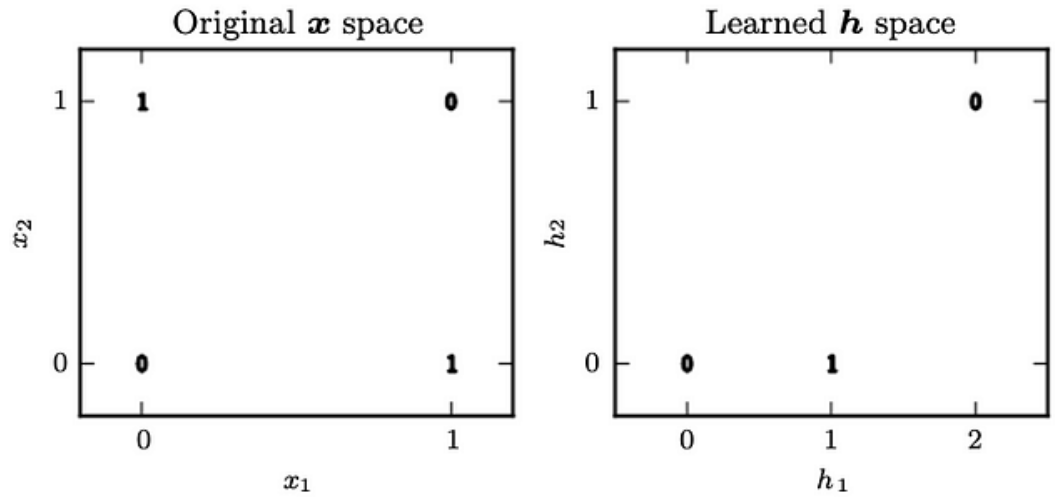
The XOR function (“exclusive or”) is an operation on two binary values, x1 and x2. When exactly one of these binary values is equal to 1, the XOR function returns 1. Otherwise, it returns 0. The XOR function provides the target function $y = f*(x)$ that we want to learn. Our model provides a function $y = f(x;\theta)$ and our learning algorithm will adapt the parameters θ to make f as similar as possible In this simple example, we will not be concerned with statistical generalization. We want our network to perform correctly on the four points X = {[0, 0], [0,1],[1,0], and [1,1]}. We will train the network on all four of these points. The only challenge is to fit the training set. We can treat this problem as a regression problem and use a mean squared error loss function. We choose this loss function to simplify the math for this example as much as possible. In practical applications, MSE is usually not an appropriate cost function for modeling binary data. More appropriate approaches a

Evaluated on our whole training set, the MSE loss function is a linear model, with θ consisting of w and b. Our model is defined to be

f (x; w, b) = x□w + b.

We can minimize J(θ) in closed form with respect to w and b using the normal equations.

$$J(\boldsymbol{\theta}) = \frac{1}{4} \sum_{\boldsymbol{x} \in \mathbb{X}} (f^*(\boldsymbol{x}) - f(\boldsymbol{x}; \boldsymbol{\theta}))^2 .$$



Solving the XOR problem by learning a representation. The bold numbers printed on the plot indicate the value that the learned function must output at each point. (Left)A linear model applied directly to the original input cannot implement the XOR function. When x1 = 0, the model's output must increase as x2 increases. When x1 = 1, the model's output must decrease as x2 increases. A linear model must apply a fixed coefficient w2 to x2. The linear model therefore cannot use the value of x1 to change the coefficient on x2 and cannot solve this problem. (Right)In the transformed space represented by the features extracted by a neural network, a linear model can now solve the problem. In our example solution, the two points that must have output 1 have been collapsed into a single point in feature space. In other

words, the nonlinear features have mapped both x = [1,0] and x = [0,1] to a single point in feature space, h = [1,0]. The linear model can now describe the function as increasing in h1 and decreasing in h2. In this example, the motivation for learning the feature space is only to make the model capacity greater so that it can fit the training set. In more realistic applications, learned representations can also help the model to **generalize**.

The following table shows the truth table for the XOR function:

Input A	Input B	Output (A XOR B)
0	0	0
0	1	1
1	0	1
1	1	0

Gradient-Based Learning:

Gradient Descent is known as one of the most commonly used optimization algorithms to train machine learning models by means of minimizing errors between actual and expected results. Further, gradient descent is also used to train Neural Networks.

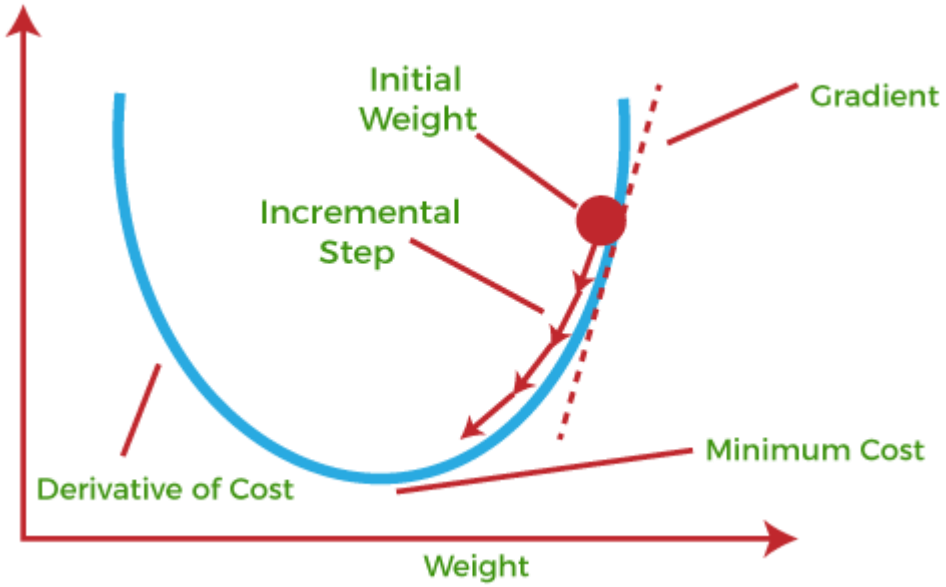
In mathematical terminology, Optimization algorithm refers to the task of minimizing/maximizing an objective function f(x) parameterized by x. Similarly, in machine learning, optimization is the task of minimizing the cost function parameterized by the model's parameters. The main objective of gradient descent is to minimize the convex function using iteration of parameter updates. Once these machine learning models are optimized, these models can be used as powerful tools for Artificial Intelligence and various computer science applications.

What is Gradient Descent or Steepest Descent?

Gradient descent was initially discovered by **"Augustin-Louis Cauchy"** in mid of 18th century. ***Gradient Descent is defined as one of the most commonly used iterative optimization algorithms of machine learning to train the machine learning and deep learning models. It helps in finding the local minimum of a function.***

The best way to define the local minimum or local maximum of a function using gradient descent is as follows:

- If we move towards a negative gradient or away from the gradient of the function at the current point, it will give the **local minimum** of that function.
- Whenever we move towards a positive gradient or towards the gradient of the function at the current point, we will get the **local maximum** of that function.



This entire procedure is known as Gradient Ascent, which is also known as steepest descent. The main objective of using a gradient descent algorithm is to minimize the cost function using iteration. To achieve this goal, it performs two steps iteratively:
Calculates the first-order derivative of the function to compute the gradient or slope of that function.
Move away from the direction of the gradient, which means slope increased from the current point by alpha times, where Alpha is defined as Learning Rate. It is a tuning parameter in the optimization process which helps to decide the length of the steps.

What is Cost-function?

The cost function is defined as the measurement of difference or error between actual values and expected values at the current position and present in the form of a single real number. It helps to increase and improve machine learning efficiency by providing feedback to this model so that it can minimize error and find the local or global minimum. Further, it continuously iterates along the direction of the negative gradient until the cost function approaches zero. At this steepest descent point, the model will stop learning further. Although cost function and loss function are considered synonymous, also there is a minor difference between them. The slight difference between the loss function and the cost function is about the error within the training of machine learning models, as loss function refers to the error of one training example, while a cost function calculates the average error across an entire training set.

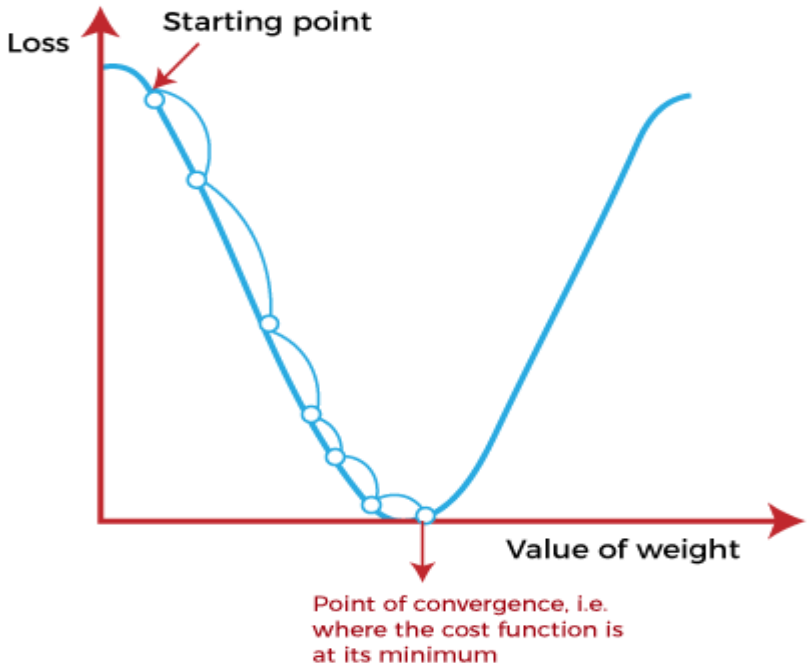
The cost function is calculated after making a hypothesis with initial parameters and modifying these parameters using gradient descent algorithms over known data to reduce the cost function.

How does Gradient Descent work?

Before starting the working principle of gradient descent, we should know some basic concepts to find out the slope of a line from linear regression. The equation for simple linear regression is given as:

1. $Y=mX+c$

Where 'm' represents the slope of the line, and 'c' represents the intercepts on the y-axis.



he starting point(shown in above fig.) is used to evaluate the performance as it is considered just as an arbitrary point. At this starting point, we will derive the first derivative or slope and then use a tangent line to calculate the steepness of this slope. Further, this slope will inform the updates to the parameters (weights and bias).

The slope becomes steeper at the starting point or arbitrary point, but whenever new parameters are generated, then steepness gradually reduces, and at the lowest point, it approaches the lowest point, which is called **a point of convergence.**

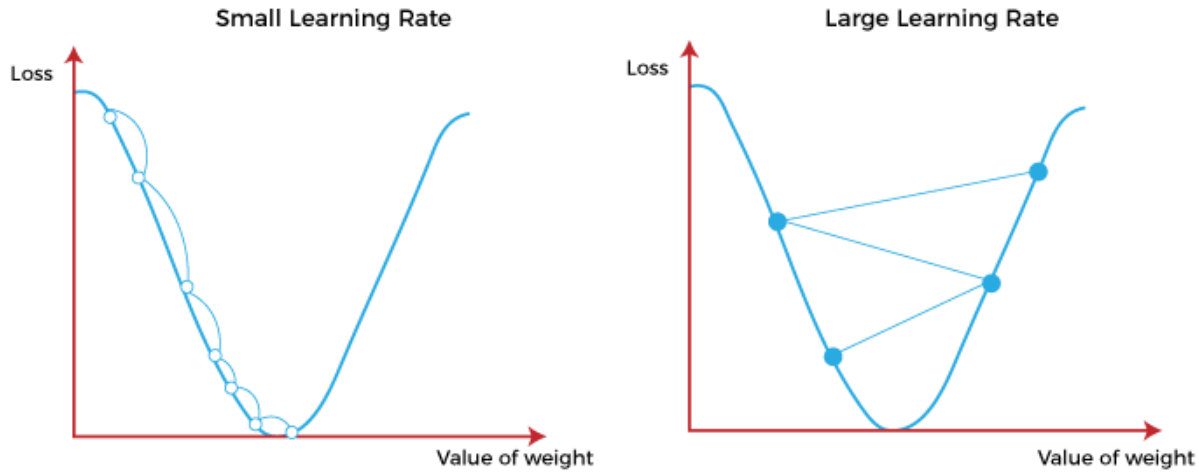
The main objective of gradient descent is to minimize the cost function or the error between expected and actual. To minimize the cost function, two data points are require

- **Direction & Learning Rate**

These two factors are used to determine the partial derivative calculation of future iteration and allow it to the point of convergence or local minimum or global minimum. Let's discuss learning rate factors in brief;

Learning Rate:

It is defined as the step size taken to reach the minimum or lowest point. This is typically a small value that is evaluated and updated based on the behavior of the cost function. If the learning rate is high, it results in larger steps but also leads to risks of overshooting the minimum. At the same time, a low learning rate shows the small step sizes, which compromises overall efficiency but gives the advantage of more precision.



Types of Gradient Descent

Based on the error in various training models, the Gradient Descent learning algorithm can be divided into **Batch gradient descent, stochastic gradient descent, and mini-batch gradient descent.** Let's understand these different types of gradient descent: 1. Batch Gradient Descent:

Batch gradient descent (BGD) is used to find the error for each point in the training set and update the model after evaluating all training examples. This procedure is known as the training epoch. In simple words, it is a greedy approach where we have to sum over all examples for each update.

Advantages of Batch gradient descent:

It produces less noise in comparison to other gradient descent.

It produces stable gradient descent convergence.

It is Computationally efficient as all resources are used for all training samples.

2. **Stochastic gradient descent**

Stochastic gradient descent (SGD) is a type of gradient descent that runs one training example per iteration. Or in other words, it processes a training epoch for each example within a dataset and updates each training example's parameters one at a time. As it requires only one training example at a time, hence it is easier to store in allocated memory. However, it shows some computational efficiency losses in comparison to batch gradient systems as it shows frequent updates that require more detail and speed. Further, due to frequent updates, it is also treated as a noisy gradient. However, sometimes it can be helpful in finding the global minimum and also escaping the local minimum.

Advantages of Stochastic gradient descent:

In Stochastic gradient descent (SGD), learning happens on every example, and it consists of a few advantages over other gradient descent.

- It is easier to allocate in desired memory.
- It is relatively fast to compute than batch gradient descent.
- It is more efficient for large datasets.

3. MiniBatch Gradient Descent:

Mini Batch gradient descent is the combination of both batch gradient descent and stochastic gradient descent. It divides the training datasets into small batch sizes then performs the updates on those batches separately. Splitting training datasets into smaller batches make a balance to maintain the computational efficiency of batch gradient descent and speed of stochastic gradient descent. Hence, we can achieve a special type of gradient descent with higher computational efficiency and less noisy gradient descent.

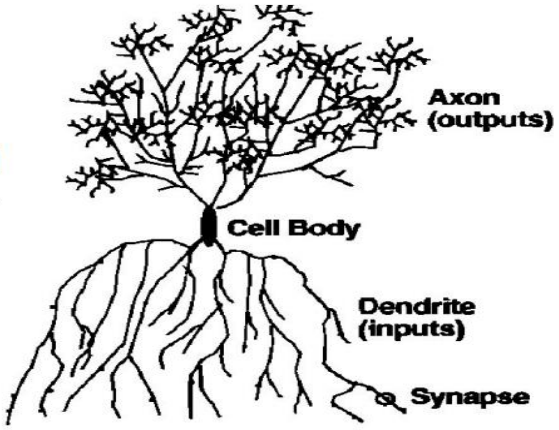
Advantages of Mini Batch gradient descent:

- It is easier to fit in allocated memory.
- It is computationally efficient.
- It produces stable gradient descent convergence.

Hidden Units:

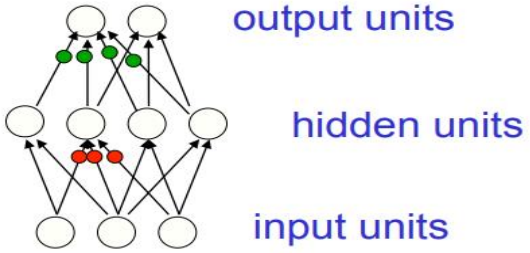
Neural networks

- Many machine learning methods inspired by biology, brains
- Our brains contain $\sim 10^{11}$ neurons, each of which communicates to $\sim 10^{4}$ other neurons
- Multi-layer perceptron, or neural network, is a popular supervised learning approach
- Defines extra functions of the inputs (hidden features), computed by neurons
- Artificial neurons called units
- Network output is linear combination of hidden units



Neural network architecture

- Network with one layer of four hidden units:



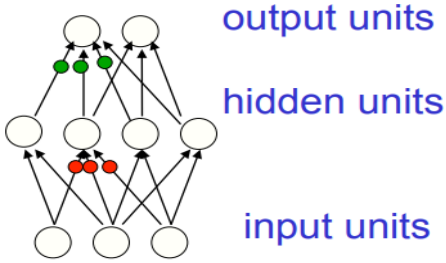
- Each unit computes value based on linear combination of values of units that point into it
- Can add more layers of hidden units: deeper hidden unit response depends on earlier hiddens

What does the network compute?

- Output of network can be written as follows, with k indexing the two output units:

$$o_k(x) = g(w_{k0} + \sum_{j=1}^{J} h_j(x) w_{kj})$$

$$h_j(x) = f(w_{j0} + \sum_{i=1}^{D} x_i v_{ji})$$



- Network with non-linear activation function *f()* is a universal approximator (esp. with increasing J)
- Standard choice of activation function: sigmoid/logistic, or tanh, or rectified linear (relu)

  tanh(z) = (exp(z) – exp(-z))/(exp(z) + exp(-z))

  relu(z) = max(0,z)

Architecture Design:

Multilayer Feed-Forward Neural Network(MFFNN) is an interconnected Artificial Neural Network with multiple layers that has neurons with weights associated with them and they compute the result using activation functions. It is one of the types of Neural Networks in

which the flow of the network is from input to output units and it does not have any loops, no feedback, and no signal moves in backward directions that is from output to hidden and input layer.

The ANN is a self-learning network that learns from sample data sets and signals, it is based on the function of the biological nervous system. The type of activation function depends on the desired output. It is a part of machine learning and AI, which are the fastest-growing fields, and lots of research is going on to make it more effective.

The Architecture of the Multilayer Feed-Forward Neural Network:

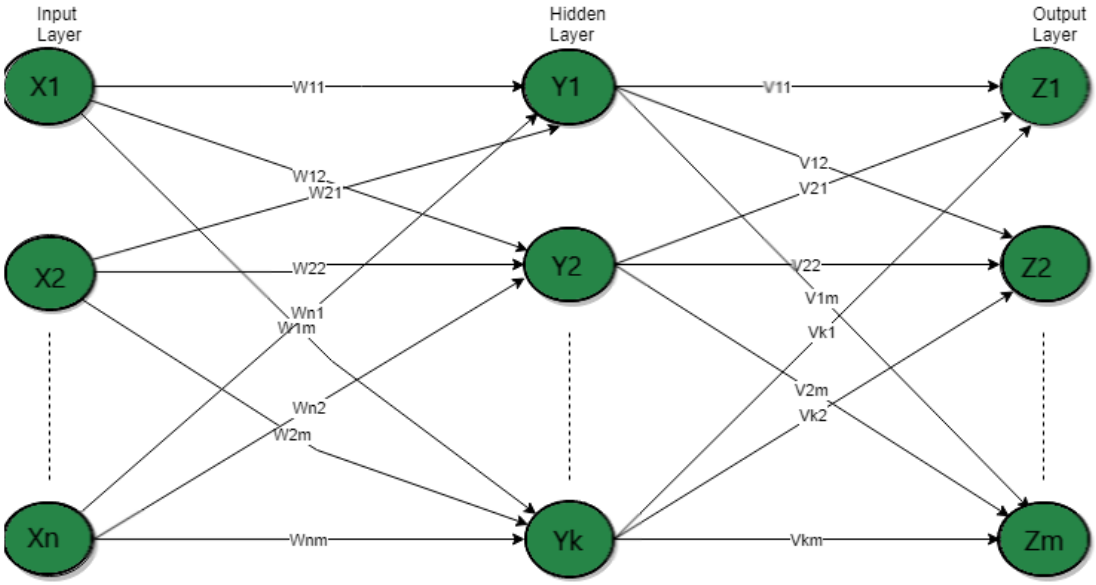
This Neural Network or Artificial Neural Network has multiple hidden layers that make it a multilayer neural Network and it is feed-forward because it is a network that follows a top-down approach to train the network. In this network there are the following layers:

Input Layer: It is starting layer of the network that has a weight associated with the signals.

Hidden Layer: This layer lies after the input layer and contains multiple neurons that perform all computations and pass the result to the output unit.

Output Layer: It is a layer that contains output units or neurons and receives processed data from the hidden layer, if there are further hidden layers connected to it then it passes the weighted unit to the connected hidden layer for further processing to get the desired result.

The input and hidden layers use sigmoid and linear activation functions whereas the output layer uses a Heaviside step activation function at nodes because it is a two-step activation function that helps in predicting results as per requirements. All units also known as neurons have weights and calculation at the hidden layer is the summation of the dot product of all weights and their signals and finally the sigmoid function of the calculated sum. Multiple hidden and output layer increases the accuracy of the output.



Application of Multilayer Feed-Forward Neural Network:

- Medical field
- Speech regeneration
- Data processing and compression
- Image processing

Limitations:

This ANN is a basic form of Neural Network that has no cycles and computes only in the forward direction. It has some limitations like sometimes information about the neighborhood is lost and in that case, it becomes difficult to process further all steps are needed to be performed again and it does not support back propagation so the network cannot learn or correct the fault of the previous stage.

Back-Propagation and other Differentiation Algorithms:

ackpropagation, or backward propagation of errors, is an algorithm that is designed to test for errors working back from output nodes to input nodes. It is an important mathematical tool for improving the accuracy of predictions in data mining and machine learning. Essentially, backpropagation is an algorithm used to calculate derivatives quickly.

There are two leading types of backpropagation networks:

Static backpropagation. Static backpropagation is a network developed to map static inputs for static outputs. Static backpropagation networks can solve static classification problems, such as optical character recognition (OCR).

Recurrent backpropagation. The recurrent backpropagation network is used for fixed-point learning. Recurrent backpropagation activation feeds forward until it reaches a fixed value.

What is a backpropagation algorithm in a neural network?

Artificial neural networks use backpropagation as a learning algorithm to compute a gradient descent with respect to weight values for the various inputs. By comparing desired outputs to achieved system outputs, the systems are tuned by adjusting connection weights to narrow the difference between the two as much as possible.

The algorithm gets its name because the weights are updated backward, from output to input.

The advantages of using a backpropagation algorithm are as follows:

- It does not have any parameters to tune except for the number of inputs.
- It is highly adaptable and efficient and does not require any prior knowledge about the network.
- It is a standard process that usually works well.
- It is user-friendly, fast and easy to program.
- Users do not need to learn any special functions.

The disadvantages of using a backpropagation algorithm are as follows:

- It prefers a matrix-based approach over a mini-batch approach.
- Data mining is sensitive to noise and irregularities.
- Performance is highly dependent on input data.
- Training is time- and resource-intensive

What is the objective of a backpropagation algorithm?

Backpropagation algorithms are used extensively to train feedforward neural networks in areas such as deep learning. They efficiently compute the gradient of the loss function with respect to the network weights. This approach eliminates the inefficient process of directly

computing the gradient with respect to each individual weight. It enables the use of gradient methods, like gradient descent or stochastic gradient descent, to train multilayer networks and update weights to minimize loss.

What is a backpropagation algorithm in machine learning?

Backpropagation requires a known, desired output for each input value in order to calculate the loss function gradient -- how a prediction differs from actual results -- as a type of supervised machine learning. Along with classifiers such as Naïve Bayesian filters and decision trees, the backpropagation training algorithm has emerged as an important part of machine learning applications that involve predictive analytics.

What is the time complexity of a backpropagation algorithm?

The time complexity of each iteration -- how long it takes to execute each statement in an algorithm -- depends on the network's structure. For multilayer perceptron, matrix multiplications dominate time.