SEMESTER IV

(UNIT 4a)

- **Supervised Machine Learning**

Supervised learning is the types of machine learning in which machines are trained using well "labeled" training data, and on basis of that data, machines predict the output. The labeled data means some input data is already tagged with the correct output.

In supervised learning, the training data provided to the machines work as the supervisor that teaches the machines to predict the output correctly. It applies the same concept as a student learns in the supervision of the teacher.

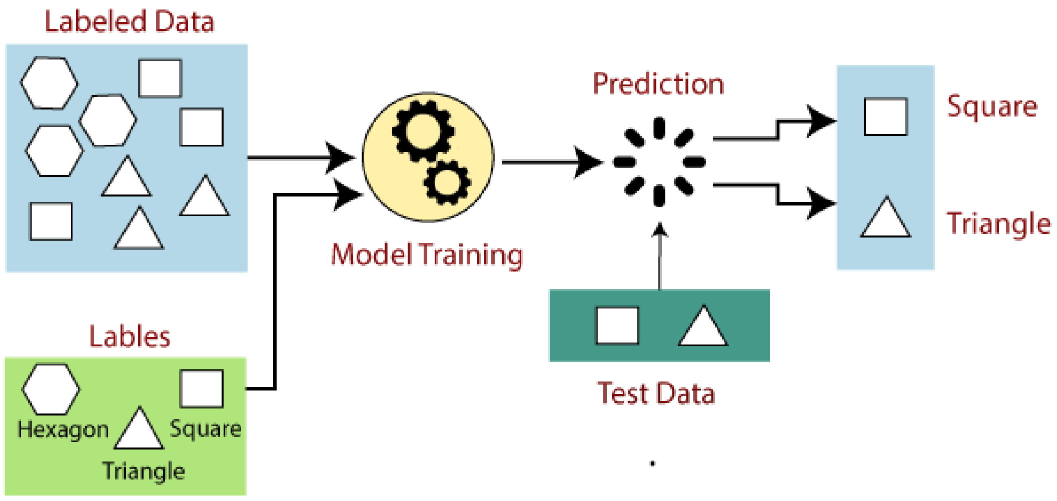
Supervised learning is a process of providing input data as well as correct output data to the machine learning model. The aim of a supervised learning algorithm is to find a mapping function to map the input variable(x) with the output variable(y).

In the real-world, supervised learning can be used for Risk Assessment, Image classification, Fraud Detection, spam filtering, etc.

How Supervised Learning Works?

In supervised learning, models are trained using labeled dataset, where the model learns about each type of data. Once the training process is completed, the model is tested on the basis of test data (a subset of the training set), and then it predicts the output.

The working of Supervised learning can be easily understood by the below example and diagram:



Suppose we have a dataset of different types of shapes which includes square, rectangle, triangle, and Polygon. Now the first step is that we need to train the model for each shape.

- If the given shape has four sides, and all the sides are equal, then it will be labelled as a Square.
- If the given shape has three sides, then it will be labeled as a triangle.
- If the given shape has six equal sides then it will be labeled as hexagon.

Now, after training, we test our model using the test set, and the task of the model is to identify the shape.

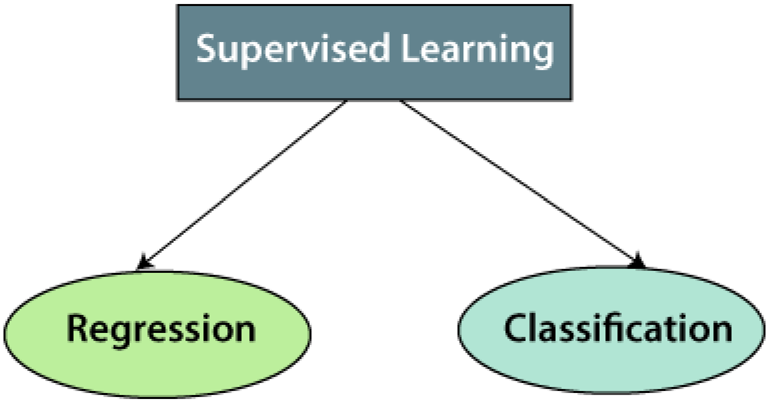
The machine is already trained on all types of shapes, and when it finds a new shape, it classifies the shape on the bases of a number of sides, and predicts the output.

Steps Involved in Supervised Learning:

- First Determine the type of training dataset
- Collect/Gather the labeled training data.
- Split the training dataset into training dataset, test dataset, and validation dataset.
- Determine the input features of the training dataset, which should have enough knowledge so that the model can accurately predict the output.
- Determine the suitable algorithm for the model, such as support vector machine, decision tree, etc.
- Execute the algorithm on the training dataset. Sometimes we need validation sets as the control parameters, which are the subset of training datasets.
- Evaluate the accuracy of the model by providing the test set. If the model predicts the correct output, which means our model is accurate.

Types of supervised Machine learning Algorithms:

Supervised learning can be further divided into two types of problems:



1. Regression
Regression algorithms are used if there is a relationship between the input variable and the output variable. It is used for the prediction of continuous variables, such as Weather forecasting, Market Trends, etc. Below are some popular Regression algorithms which come under supervised learning:

- Linear Regression
- Regression Trees
- Non-Linear Regression
- Bayesian Linear Regression
- Polynomial Regression

2. Classification
Classification algorithms are used when the output variable is categorical, which means there are two classes such as Yes-No, Male-Female, True-false, etc.
Spam Filtering,

- Random Forest
- Decision Trees
- Logistic Regression
- Support vector Machines

Advantages of Supervised learning:

- With the help of supervised learning, the model can predict the output on the basis of prior experiences.
- In supervised learning, we can have an exact idea about the classes of objects.
- Supervised learning model helps us to solve various real-world problems such as fraud detection, spam filtering, etc.

Disadvantages of supervised learning:

- ○ Supervised learning models are not suitable for handling the complex tasks.
- ○ Supervised learning cannot predict the correct output if the test data is different from the training dataset.
- ○ Training required lots of computation times.
- ○ In supervised learning, we need enough knowledge about the classes of object.
- Bayesian Learning :-

Features of Bayesian learning methods:

• Each observed training example can incrementally decrease or increase the estimated probability that a hypothesis is correct.

– This provides a more flexible approach to learning than algorithms that completely eliminate a hypothesis if it is found to be inconsistent with any single example.

• Prior knowledge can be combined with observed data to determine the final probability of a hypothesis. In Bayesian learning, prior knowledge is provided by asserting

– a prior probability for each candidate hypothesis, and

– a probability distribution over observed data for each possible hypothesis.

• Bayesian methods can accommodate hypotheses that make probabilistic predictions

• New instances can be classified by combining the predictions of multiple hypotheses, weighted by their probabilities.

• Even in cases where Bayesian methods prove computationally intractable, they can provide a standard of optimal decision making against which other practical methods can be measured.

Difficulties with Bayesian Methods:-

• Require initial knowledge of many probabilities

– When these probabilities are not known in advance they are often estimated based on background knowledge, previously available data, and assumptions about the form of the underlying distributions.

• Significant computational cost is required to determine the Bayes optimal hypothesis in the general case (linear in the number of candidate hypotheses). – In certain specialized situations, this computational cost can be significantly reduced.

Bayes Theorem

• In machine learning, we try to determine the best hypothesis from some hypothesis space H, given the observed training data D.

• In Bayesian learning, the best hypothesis means the most probable hypothesis, given the data D plus any initial knowledge about the prior probabilities of the various hypotheses in H.

• Bayes theorem provides a way to calculate the probability of a hypothesis based on its prior probability, the probabilities of observing various data given the hypothesis, and the observed data itself.

Bayes Theorem

- P(h) is prior probability of hypothesis h

– P(h) to denote the initial probability that hypothesis h holds, before observing training data.

– P(h) may reflect any background knowledge we have about the chance that h is correct. If we have no such prior knowledge, then each candidate hypothesis might simply get the same prior probability.

- P(D) is prior probability of training data D

– The probability of D given no knowledge about which hypothesis holds

- P(h|D) is posterior probability of h given D

– P(h|D) is called the posterior probability of h, because it reflects our confidence that h holds after we have seen the training data D.

– The posterior probability P(h|D) reflects the influence of the training data D, in contrast to the prior probability P(h), which is independent of D.

- P(D|h) is posterior probability of D given h

– The probability of observing data D given some world in which hypothesis h holds.

– Generally, we write P(xly) to denote the probability of event x given event y.

Bayes Theorem

• In ML problems, we are interested in the probability P(h|D) that h holds given the observed training data D.

• Bayes theorem provides a way to calculate the posterior probability P(h|D), from the prior probability P(h), together with P(D) and P(D|h).

Bayes Theorem: $P(h|D) = P(D|h)\ P(h) / P(D)$

• P(h|D) increases with P(h) and P(D|h) according to Bayes theorem.

• P(h|D) decreases as P(D) increases, because the more probable it is that D will be observed independent of h, the less evidence D provides in support of h.

- Naïve Bayes Classifier Algorithm:-

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.
- It is mainly used in text classification that includes a high-dimensional training dataset.
- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.
- Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.

Why is it called Naïve Bayes?

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:

- Naïve: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.
- Bayes: It is called Bayes because it depends on the principle of Bayes' Theorem.

Bayes' Theorem:

- Bayes' theorem is also known as Bayes' Rule or Bayes' law, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.
- The formula for Bayes' theorem is given as:

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

Where,
P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.
P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.
P(A) is Prior Probability: Probability of hypothesis before observing the evidence.
P(B) is Marginal Probability: Probability of Evidence.

Working of Naïve Bayes' Classifier:

Working of Naïve Bayes' Classifier can be understood with the help of the below example:
Suppose we have a dataset of weather conditions and corresponding target variable "Play". So using this dataset we need to decide that whether we should play or not on a particular day according to the weather conditions. So to solve this problem, we need to follow the below steps:

1. Convert the given dataset into frequency tables.
2. Generate Likelihood table by finding the probabilities of given features.
3. Now, use Bayes theorem to calculate the posterior probability.

Problem: If the weather is sunny, then the Player should play or not?
Solution: To solve this, first consider the below dataset:

	Outlook	Play
0	Rainy	Yes
1	Sunny	Yes
2	Overcast	Yes
3	Overcast	Yes
4	Sunny	No
5	Rainy	Yes
6	Sunny	Yes

7	Overcast	Yes
8	Rainy	No
9	Sunny	No
10	Sunny	Yes
11	Rainy	No
12	Overcast	Yes
13	Overcast	Yes

Frequency table for the Weather Conditions:

Weather	Yes	No
Overcast	5	0
Rainy	2	2
Sunny	3	2
Total	10	5

Likelihood table weather condition:

Weather	No	Yes	
Overcast	0	5	5/14= 0.35
Rainy	2	2	4/14=0.29

Sunny	2	3	5/14=0.35
All	4/14=0.29	10/14=0.71	

Applying Bayes'theorem:
P(Yes|Sunny)= P(Sunny|Yes)*P(Yes)/P(Sunny)
P(Sunny|Yes)= 3/10= 0.3
P(Sunny)= 0.35
P(Yes)=0.71
So P(Yes|Sunny) = 0.3*0.71/0.35= 0.60
P(No|Sunny)= P(Sunny|No)*P(No)/P(Sunny)
P(Sunny|NO)= 2/4=0.5
P(No)= 0.29
P(Sunny)= 0.35
So P(No|Sunny)= 0.5*0.29/0.35 = 0.41
So as we can see from the above calculation that P(Yes|Sunny)>P(No|Sunny)
Hence on a Sunny day, Player can play the game.

Advantages of Naïve Bayes Classifier:

- Naïve Bayes is one of the fast and easy ML algorithms to predict a class of datasets.
- It can be used for Binary as well as Multi-class Classifications.
- It performs well in Multi-class predictions as compared to the other Algorithms.
- It is the most popular choice for text classification problems.

Disadvantages of Naïve Bayes Classifier:

- Naive Bayes assumes that all features are independent or unrelated, so it cannot learn the relationship between features.

Applications of Naïve Bayes Classifier:

- It is used for Credit Scoring.
- It is used in medical data classification.

- ○ It can be used in real-time predictions because Naïve Bayes Classifier is an eager learner.
- ○ It is used in Text classification such as Spam filtering and Sentiment analysis.

- Decision Tree in Machine Learning

A decision tree in machine learning is a versatile, interpretable algorithm used for predictive modelling. It structures decisions based on input data, making it suitable for both classification and regression tasks. This article delves into the components, terminologies, construction, and advantages of decision trees, exploring their applications and learning algorithms.

Decision Tree in Machine Learning

A decision tree is a type of supervised learning algorithm that is commonly used in machine learning to model and predict outcomes based on input data. It is a tree-like structure where each internal node tests on attribute, each branch corresponds to attribute value and each leaf node represents the final decision or prediction. The decision tree algorithm falls under the category of supervised learning. They can be used to solve both regression and classification problems.

Decision Tree Terminologies

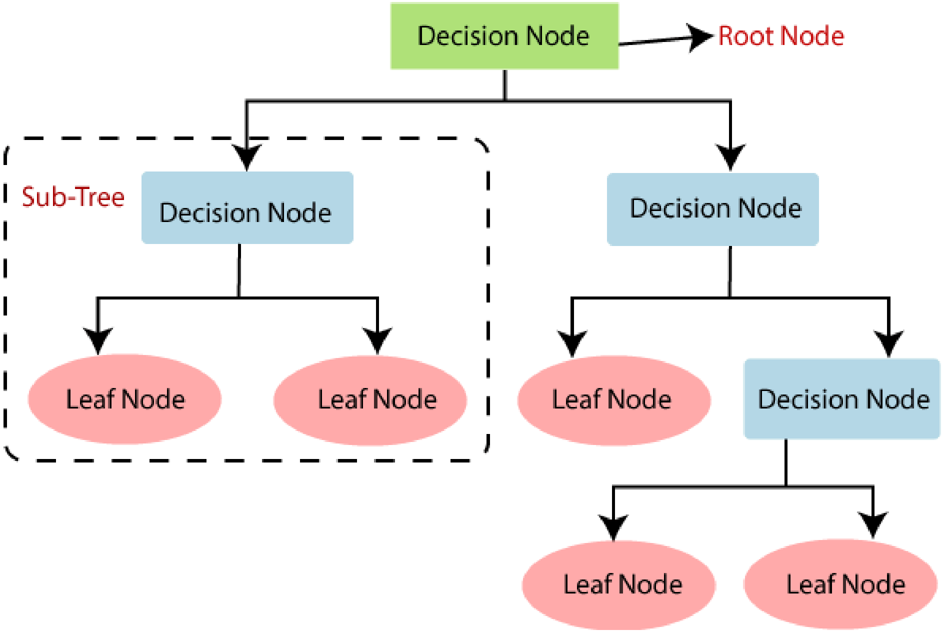
There are specialized terms associated with decision trees that denote various components and facets of the tree structure and decision-making procedure. :

- Root Node: A decision tree's root node, which represents the original choice or feature from which the tree branches, is the highest node.
- Internal Nodes (Decision Nodes): Nodes in the tree whose choices are determined by the values of particular attributes. There are branches on these nodes that go to other nodes.
- Leaf Nodes (Terminal Nodes): The branches' termini, when choices or forecasts are decided upon. There are no more branches on leaf nodes.
- Branches (Edges): Links between nodes that show how decisions are made in response to particular circumstances.
- Splitting: The process of dividing a node into two or more sub-nodes based on a decision criterion. It involves selecting a feature and a threshold to create subsets of data.
- Parent Node: A node that is split into child nodes. The original node from which a split originates.
- Child Node: Nodes created as a result of a split from a parent node.
- Decision Criterion: The rule or condition used to determine how the data should be split at a decision node. It involves comparing feature values against a threshold.
- Pruning: The process of removing branches or nodes from a decision tree to improve its generalisation and prevent overfitting.

Understanding these terminologies is crucial for interpreting and working with decision trees in machine learning applications.

How Decision Tree is formed?

The process of forming a decision tree involves recursively partitioning the data based on the values of different attributes. The algorithm selects the best attribute to split the data at each internal node, based on certain criteria such as information gain or Gini impurity. This splitting process continues until a stopping criterion is met, such as reaching a maximum depth or having a minimum number of instances in a leaf node.



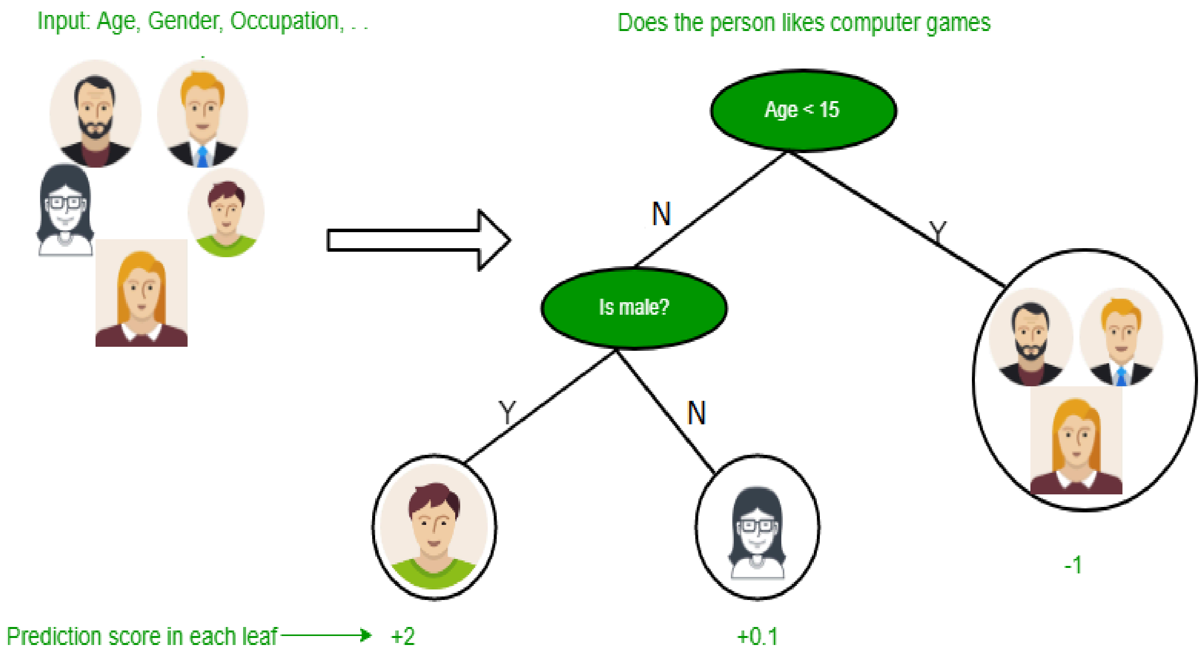
Why Decision Tree?

Decision trees are widely used in machine learning for a number of reasons:

- Decision trees are so versatile in simulating intricate decision-making processes, because of their interpretability and versatility.
- Their portrayal of complex choice scenarios that take into account a variety of causes and outcomes is made possible by their hierarchical structure.
- They provide comprehensible insights into the decision logic, decision trees are especially helpful for tasks involving categorisation and regression.
- They are proficient with both numerical and categorical data, and they can easily adapt to a variety of datasets thanks to their autonomous feature selection capability.
- Decision trees also provide simple visualization, which helps to comprehend and elucidate the underlying decision processes in a model.

Decision Tree Approach

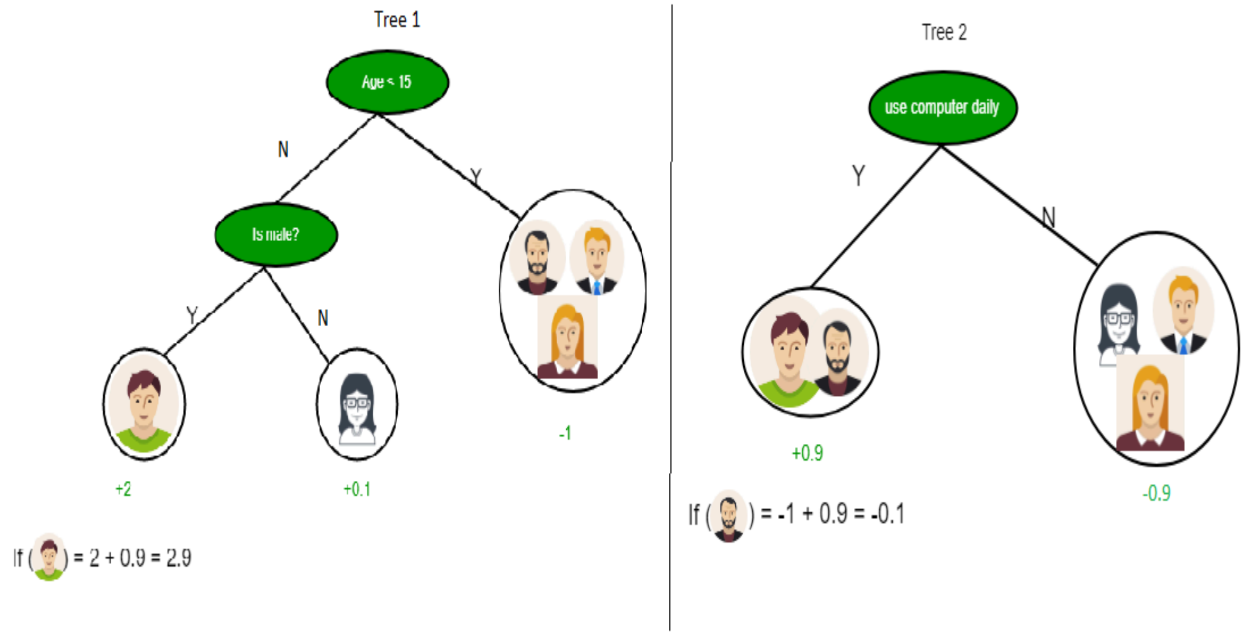
Decision tree uses the tree representation to solve the problem in which each leaf node corresponds to a class label and attributes are represented on the internal node of the tree. We can represent any boolean function on discrete attributes using the decision tree.


Input: Age, Gender, Occupation, . .
Does the person likes computer games
Age < 15
N
Y
Is male?
Y
N
-1
Prediction score in each leaf
+2
+0.1

Below are some assumptions that we made while using the decision tree:

At the beginning, we consider the whole training set as the root.

- Feature values are preferred to be categorical. If the values are continuous then they are discretized prior to building the model.
- On the basis of attribute values, records are distributed recursively.
- We use statistical methods for ordering attributes as root or the internal node.



As you can see from the above image the Decision Tree works on the Sum of Product form which is also known as Disjunctive Normal Form. In the above image, we are predicting the use of computer in the daily life of people. In the Decision Tree, the major challenge is the identification of the attribute for the root node at each level. This process is known as attribute selection. We have two popular attribute selection measures:

1. Information Gain
2. Gini Index

1. Information Gain:

When we use a node in a decision tree to partition the training instances into smaller subsets the entropy changes. Information gain is a measure of this change in entropy.

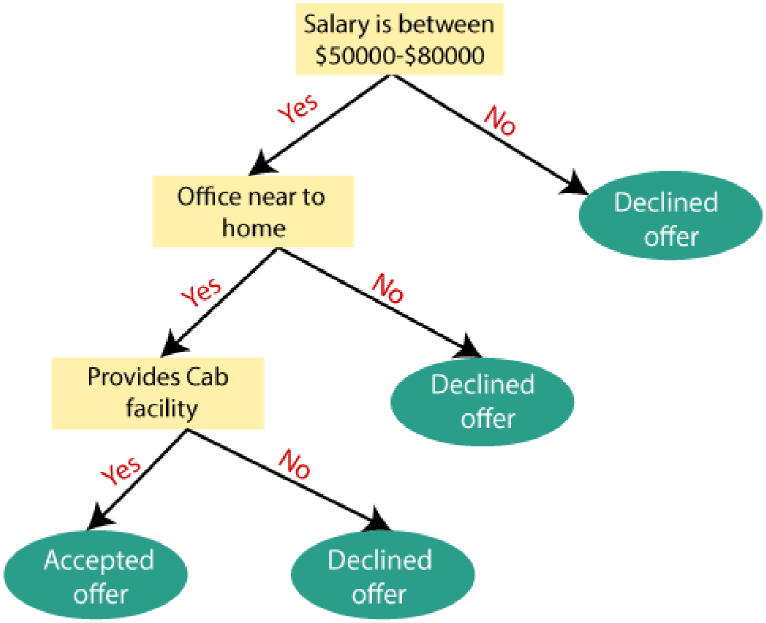
- Suppose S is a set of instances,
- A is an attribute
- Sv is the subset of S
- v represents an individual value that the attribute A can take and Values (A) is the set of all possible values of A, then
- Gain(S,A) = Entropy(S) - vA$|S_{v|}|S|$. Entropy (S_v)

Entropy: is the measure of uncertainty of a random variable, it characterizes the impurity of an arbitrary collection of examples. The higher the entropy more the information content.

Suppose S is a set of instances, A is an attribute, Sv is the subset of S with A = v, and Values (A) is the set of all possible values of A.

- Gain(S,A) = Entropy(S) - $_{v€\ \mathrm{values}(A)}$ $|S_{v|}|S|$. Entropy (S_v)

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



Example:
For the set X = {a,a,a,b,b,b,b,b}
Total instances: 8
Instances of b: 5
Instances of a: 3
Entropy H (X)=$[(38\log_{2}{}_{38} + 58\log_{2}{}_{58})]$
$=-[0.375(-1.415)+0.625(-0.678)]$
$=-(-0.53-0.424)$
$=0.954$
Building Decision Tree using Information Gain the essentials:

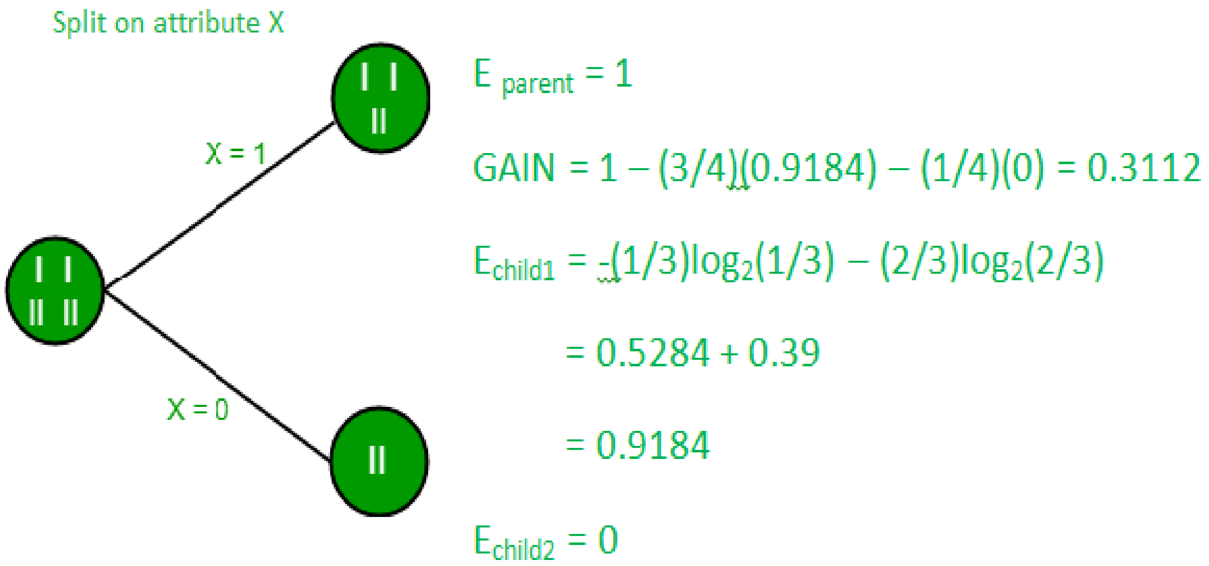
- Start with all training instances associated with the root node
- Use info gain to choose which attribute to label each node with
- Note: No root-to-leaf path should contain the same discrete attribute twice
- Recursively construct each subtree on the subset of training instances that would be classified down that path in the tree.
- If all positive or all negative training instances remain, the label that node “yes” or “no” accordingly
- If no attributes remain, label with a majority vote of training instances left at that node

- If no instances remain, label with a majority vote of the parent’s training instances.

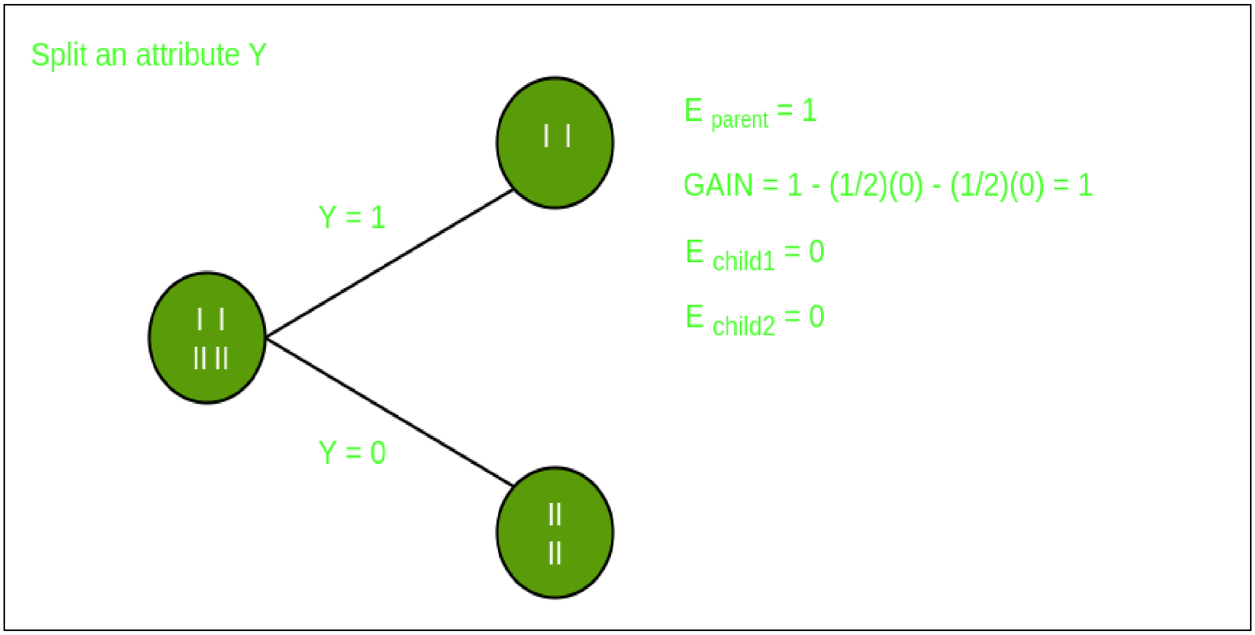
Example: Now, let us draw a Decision Tree for the following data using Information gain.
Training set: 3 features and 2 classes

X	Y	Z	C
1	1	1	I
1	1	0	I
0	0	1	II
1	0	0	II

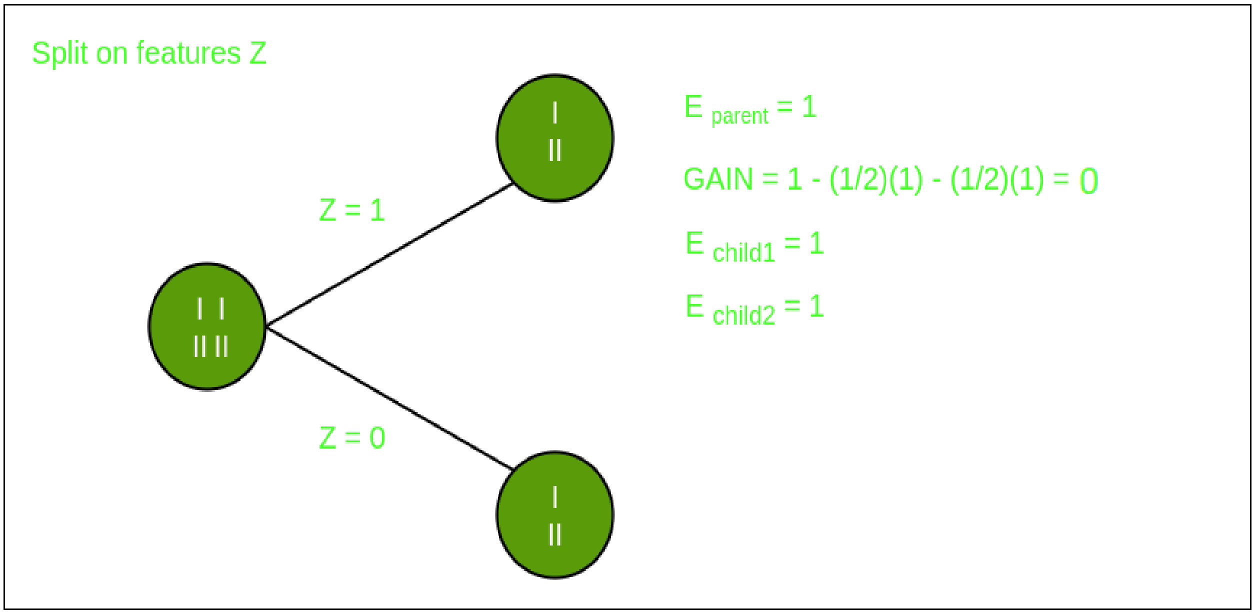
Here, we have 3 features and 2 output classes. To build a decision tree using Information gain. We will take each of the features and calculate the information for each feature.


Split on attribute X
X = 1
X = 0
E parent = 1
GAIN = 1 – (3/4)(0.9184) – (1/4)(0) = 0.3112
Echild1 = -(1/3)log2(1/3) – (2/3)log2(2/3)
= 0.5284 + 0.39
= 0.9184
Echild2 = 0

Split on feature X

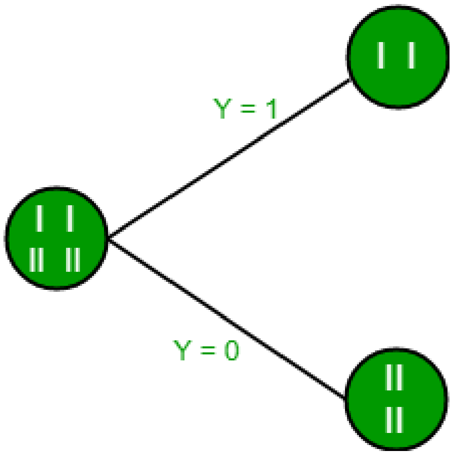


Split on feature Y



Split on feature Z

From the above images, we can see that the information gain is maximum when we make a split on feature Y. So, for the root node best-suited feature is feature Y. Now we can see that while splitting the dataset by feature Y, the child contains a pure subset of the target variable. So we don't need to further split the dataset. The final tree for the above dataset would look like this:



2. Gini Index

- Gini Index is a metric to measure how often a randomly chosen element would be incorrectly identified.
- It means an attribute with a lower Gini index should be preferred.
- Sklearn supports "Gini" criteria for Gini Index and by default, it takes "gini" value.
- The Formula for the calculation of the Gini Index is given below.

The Formula for Gini Index is given by :

$$\text{Gini}(S) = 1 - \sum_{i=1}^{c} p_i^2$$

Gini Impurity

The Gini Index is a measure of the inequality or impurity of a distribution, commonly used in decision trees and other machine learning algorithms. It ranges from 0 to 0.5, where 0 indicates a pure set (all instances belong to the same class), and 0.5 indicates a maximally impure set (instances are evenly distributed across classes).

Some additional features and characteristics of the Gini Index are:

- It is calculated by summing the squared probabilities of each outcome in a distribution and subtracting the result from 1.
- A lower Gini Index indicates a more homogeneous or pure distribution, while a higher Gini Index indicates a more heterogeneous or impure distribution.
- In decision trees, the Gini Index is used to evaluate the quality of a split by measuring the difference between the impurity of the parent node and the weighted impurity of the child nodes.
- Compared to other impurity measures like entropy, the Gini Index is faster to compute and more sensitive to changes in class probabilities.
- One disadvantage of the Gini Index is that it tends to favour splits that create equally sized child nodes, even if they are not optimal for classification accuracy.
- In practice, the choice between using the Gini Index or other impurity measures depends on the specific problem and dataset, and often requires experimentation and tuning.

Example of a Decision Tree Algorithm

Forecasting Activities Using Weather Information

- Root node: Whole dataset
- Attribute : “Outlook” (sunny, cloudy, rainy).
- Subsets: Overcast, Rainy, and Sunny.
- Recursive Splitting: Divide the sunny subset even more according to humidity, for example.
- Leaf Nodes: Activities include “swimming,” “hiking,” and “staying inside.”

Beginning with the entire dataset as the root node of the decision tree:

- Determine the best attribute to split the dataset based on information gain, which is calculated by the formula: Information gain = Entropy(parent) – [Weighted average] * Entropy(children), where entropy is a measure of impurity or disorder of a set of examples, and the weighted average is based on the number of examples in each child node.

- Create a new internal node that corresponds to the best attribute and connects it to the root node. For example, if the best attribute is “outlook” (which can have values “sunny”, “overcast”, or “rainy”), we create a new node labeled “outlook” and connect it to the root node.
- Partition the dataset into subsets based on the values of the best attribute. For example, we create three subsets: one for instances where the outlook is “sunny”, one for instances where the outlook is “overcast”, and one for instances where the outlook is “rainy”.
- Recursively repeat steps 1-4 for each subset until all instances in a given subset belong to the same class or no further splitting is possible. For example, if the subset of instances where the outlook is “overcast” contains only instances where the activity is “hiking”, we assign a leaf node labeled “hiking” to this subset. If the subset of instances where the outlook is “sunny” is further split based on the humidity attribute, we repeat steps 2-4 for this subset.
- Assign a leaf node to each subset that contains instances that belong to the same class. For example, if the subset of instances where the outlook is “rainy” contains only instances where the activity is “stay inside”, we assign a leaf node labeled “stay inside” to this subset.
- Make predictions based on the decision tree by traversing it from the root node to a leaf node that corresponds to the instance being classified. For example, if the outlook is “sunny” and the humidity is “high”, we traverse the decision tree by following the “sunny” branch and then the “high humidity” branch, and we end up at a leaf node labeled “swimming”, which is our predicted activity.
- ID3 Algorithm

A well-known decision tree approach for machine learning is the Iterative Dichotomiser 3 (ID3) algorithm. By choosing the best characteristic at each node to partition the data depending on information gain, it recursively constructs a tree. The goal is to make the final subsets as homogeneous as possible. By choosing features that offer the greatest reduction in entropy or uncertainty, ID3 iteratively grows the tree. The procedure keeps going until a halting requirement is satisfied, like a minimum subset size or a maximum tree depth.

How ID3 Works

The ID3 algorithm is specifically designed for building decision trees from a given dataset. Its primary objective is to construct a tree that best explains the relationship between attributes in the data and their corresponding class labels.

1. Selecting the Best Attribute

- ID3 employs the concept of entropy and information gain to determine the attribute that best separates the data. Entropy measures the impurity or randomness in the dataset.
- The algorithm calculates the entropy of each attribute and selects the one that results in the most significant information gain when used for splitting the data.

2. Creating Tree Nodes

- The chosen attribute is used to split the dataset into subsets based on its distinct values.
- For each subset, ID3 recurses to find the next best attribute to further partition the data, forming branches and new nodes accordingly.

3. Stopping Criteria

- The recursion continues until one of the stopping criteria is met, such as when all instances in a branch belong to the same class or when all attributes have been used for splitting.

4. Handling Missing Values

- ID3 can handle missing attribute values by employing various strategies like attribute mean/mode substitution or using majority class values.

5. Tree Pruning

- Pruning is a technique to prevent overfitting. While not directly included in ID3, post-processing techniques or variations like C4.5 incorporate pruning to improve the tree's generalization.

Advantages of Decision Tree

- Easy to understand and interpret, making them accessible to non-experts.
- Handle both numerical and categorical data without requiring extensive preprocessing.
- Provides insights into feature importance for decision-making.
- Handle missing values and outliers without significant impact.
- Applicable to both classification and regression tasks.

Disadvantages of Decision Tree

- Disadvantages include the potential for overfitting
- Sensitivity to small changes in data, limited generalization if training data is not representative
- Potential bias in the presence of imbalanced data.

- Pruning: Getting an Optimal Decision tree

Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree. A too-large tree increases the risk of overfitting, and a small tree may not capture all the important features of the dataset. Therefore, a technique that decreases the size of the learning tree without reducing accuracy is known as Pruning. There are mainly two types of tree pruning technology used:

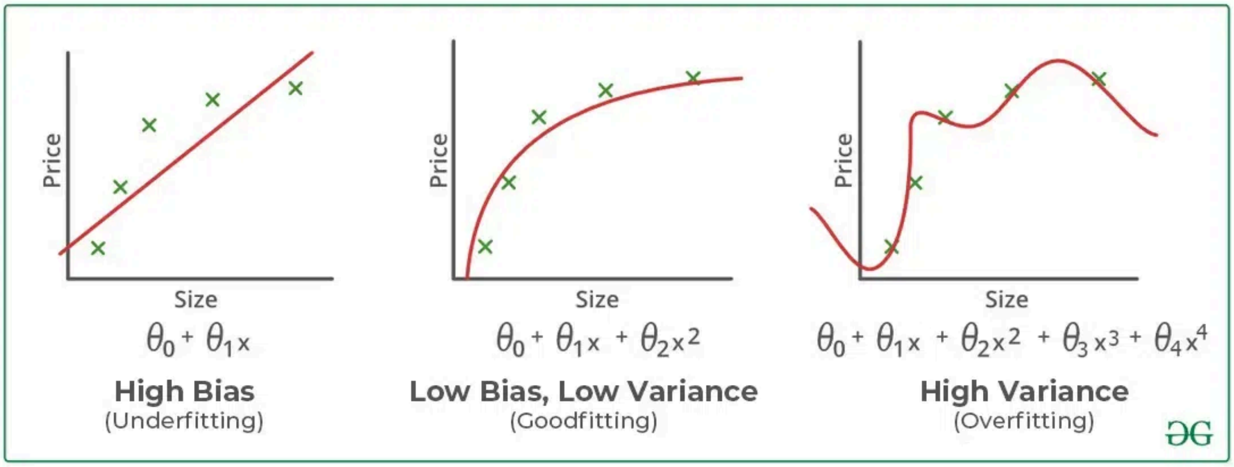
- Cost Complexity Pruning
- Reduced Error Pruning.

- Underfitting and Overfitting

When we talk about the Machine Learning model, we actually talk about how well it performs and its accuracy which is known as prediction errors. Let us consider that we are designing a machine learning model. A model is said to be a good machine learning model if it generalizes any new input data from the problem domain in a proper way. This helps us to make predictions about future data, that the data model has never seen. Now, suppose we want to check how well our machine learning model learns and generalizes to the new data. For that, we have overfitting and underfitting, which are majorly responsible for the poor performances of the machine learning algorithms.

Bias and Variance in Machine Learning

- Bias: Bias refers to the error due to overly simplistic assumptions in the learning algorithm. These assumptions make the model easier to comprehend and learn but might not capture the underlying complexities of the data. It is the error due to the model's inability to represent the true relationship between input and output accurately. When a model has poor performance both on the training and testing data means high bias because of the simple model, indicating underfitting.
- Variance: Variance, on the other hand, is the error due to the model's sensitivity to fluctuations in the training data. It's the variability of the model's predictions for different instances of training data. High variance occurs when a model learns the training data's noise and random fluctuations rather than the underlying pattern. As a result, the model performs well on the training data but poorly on the testing data, indicating overfitting.



Bias and Variance

Underfitting in Machine Learning

A statistical model or a machine learning algorithm is said to have underfitting when a model is too simple to capture data complexities. It represents the inability of the model to learn the training data effectively result in poor performance both on the training and testing data. In simple terms, an underfit model's are inaccurate, especially when applied to new, unseen examples. It mainly happens when we uses very simple model with overly simplified assumptions. To address underfitting problem of the model, we need to use more complex models, with enhanced feature representation, and less regularization.
Note: The underfitting model has High bias and low variance.

Reasons for Underfitting

1. The model is too simple, So it may be not capable to represent the complexities in the data.
2. The input features which is used to train the model is not the adequate representations of underlying factors influencing the target variable.
3. The size of the training dataset used is not enough.
4. Excessive regularization are used to prevent the overfitting, which constraint the model to capture the data well.
5. Features are not scaled.

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.

Overfitting in Machine Learning

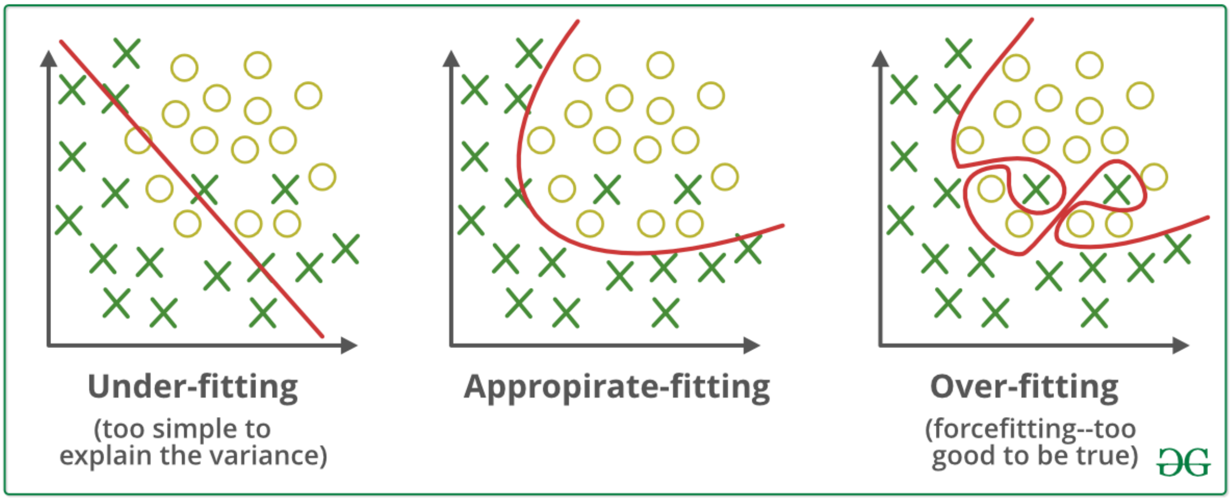
A statistical model is said to be overfitted when the model does not make accurate predictions on testing data. When a model gets trained with so much data, it starts learning from the noise and inaccurate data entries in our data set. And when testing with test data results in High variance. Then the model does not categorize the data correctly, because of too many details and noise. The causes of overfitting are the non-parametric and non-linear methods because these types of machine learning algorithms have more freedom in building the model based on the dataset and therefore they can really build unrealistic models. A solution to avoid overfitting is using a linear algorithm if we have linear data or using the parameters like the maximal depth if we are using decision trees.
In a nutshell, Overfitting is a problem where the evaluation of machine learning algorithms on training data is different from unseen data.

Reasons for Overfitting:

1. High variance and low bias.
2. The model is too complex.
3. The size of the training data.

Techniques to Reduce Overfitting

1. Improving the quality of training data reduces overfitting by focusing on meaningful patterns, mitigate the risk of fitting the noise or irrelevant features.
2. Increase the training data can improve the model's ability to generalize to unseen data and reduce the likelihood of overfitting.
3. Reduce model complexity.
4. Early stopping during the training phase (have an eye over the loss over the training period as soon as loss begins to increase stop training).
5. Ridge Regularization and Lasso Regularization.
6. Use dropout for neural networks to tackle overfitting.


Under-fitting
(too simple to explain the variance)
Appropirate-fitting
Over-fitting
(forcefitting--too good to be true)

Underfitting and Overfitting

Conclusion

Decision trees, a key tool in machine learning, model and predict outcomes based on input data through a tree-like structure. They offer interpretability, versatility, and simple visualization, making them valuable for both categorization and regression tasks. While decision trees have advantages like ease of understanding, they may face challenges such as overfitting. Understanding their terminologies and formation process is essential for effective application in diverse scenarios.

SEMESTER IV (UNIT 4B)

- What is Unsupervised Learning?

As the name suggests, unsupervised learning is a machine learning technique in which models are not supervised using training dataset. Instead, models itself find the hidden patterns and insights from the given data. It can be compared to learning which takes place in the human brain while learning new things. It can be defined as:

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.
Example: Suppose the unsupervised learning algorithm is given an input dataset containing images of different types of cats and dogs. The algorithm is never trained upon the given dataset, which means it does not have any idea about the features of the dataset. The task of the unsupervised learning algorithm is to identify the image features on their own. Unsupervised learning algorithm will perform this task by clustering the image dataset into the groups according to similarities between images.

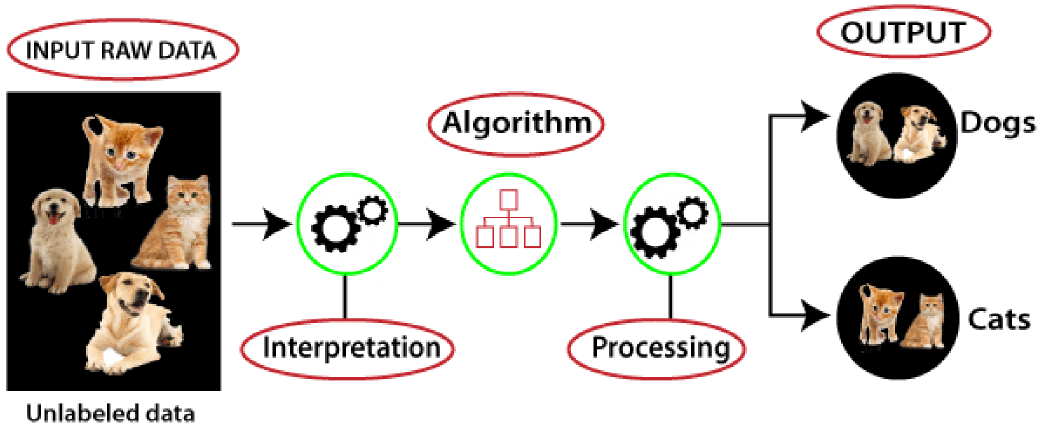
Why use Unsupervised Learning?

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:

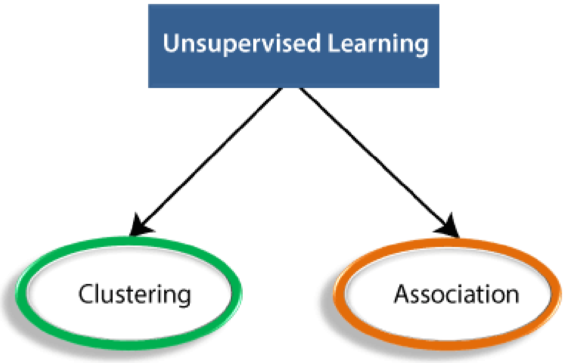


Here, we have taken an unlabeled input data, which means it is not categorized and corresponding outputs are also not given. Now, this unlabeled input data is fed to the machine learning model in order to train it. Firstly, it will interpret the raw data to find the hidden patterns from the data and 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 labeled input data.
- Unsupervised learning is preferable as it is easy to get unlabeled data in comparison to labeled 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 labeled, and algorithms do not know the exact output in advance.

Application of Unsupervised learning

Non-supervised learning can be used to solve a wide variety of problems, including:

- Anomaly detection: Unsupervised learning can identify unusual patterns or deviations from normal behavior in data, enabling the detection of fraud, intrusion, or system failures.
- Scientific discovery: Unsupervised learning can uncover hidden relationships and patterns in scientific data, leading to new hypotheses and insights in various scientific fields.
- Recommendation systems: Unsupervised learning can identify patterns and similarities in user behavior and preferences to recommend products, movies, or music that align with their interests.

- Customer segmentation: Unsupervised learning can identify groups of customers with similar characteristics, allowing businesses to target marketing campaigns and improve customer service more effectively.
- Image analysis: Unsupervised learning can group images based on their content, facilitating tasks such as image classification, object detection, and image retrieval.

Supervised vs. Unsupervised Machine Learning

Parameters	Supervised machine learning	Unsupervised machine learning
Input Data	Algorithms are trained using labeled data.	Algorithms are used against data that is not labeled
Computational Complexity	Simpler method	Computationally complex
Accuracy	Highly accurate	Less accurate
No. of classes	No. of classes is known	No. of classes is not known
Data Analysis	Uses offline analysis	Uses real-time analysis of data
Algorithms used	Linear and Logistics regression, Random forest, multi-class classification, decision tree, Support Vector Machine, Neural Network, etc.	K-Means clustering, Hierarchical clustering, KNN, Apriori algorithm, etc.

Output	Desired output is given.	Desired output is not given.
Training data	Use training data to infer model.	No training data is used.
Complex model	It is not possible to learn larger and more complex models than with supervised learning.	It is possible to learn larger and more complex models with unsupervised learning.
Model	We can test our model.	We can not test our model.
Called as	Supervised learning is also called classification.	Unsupervised learning is also called clustering.
Example	Example: Optical character recognition.	Example: Find a face in an image.
Supervision	supervised learning needs supervision to train the model.	Unsupervised learning does not need any supervision to train the model.

- Clustering in Machine Learning

Clustering or cluster analysis is a machine learning technique, which groups the unlabelled dataset. It can be defined as "A way of grouping the data points into different clusters, consisting of similar data points. The objects with the possible similarities remain in a group that has less or no similarities with another group."

It does it by finding some similar patterns in the unlabelled dataset such as shape, size, color, behavior, etc., and divides them as per the presence and absence of those similar patterns.

It is an unsupervised learning method, hence no supervision is provided to the algorithm, and it deals with the unlabeled dataset.

After applying this clustering technique, each cluster or group is provided with a cluster-ID. ML system can use this id to simplify the processing of large and complex datasets.

The clustering technique is commonly used for statistical data analysis.

Note: Clustering is somewhere similar to the classification algorithm, but the difference is the type of dataset that we are using. In classification, we work with the labeled data set, whereas in clustering, we work with the unlabelled dataset.

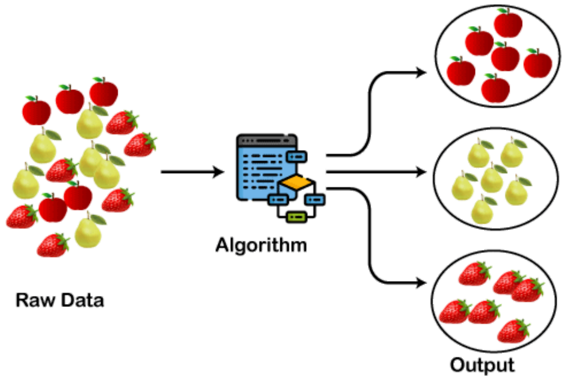
Example: Let's understand the clustering technique with the real-world example of Mall: When we visit any shopping mall, we can observe that the things with similar usage are grouped together. Such as the t-shirts are grouped in one section, and trousers are at other sections, similarly, at vegetable sections, apples, bananas, Mangoes, etc., are grouped in separate sections, so that we can easily find out the things. The clustering technique also works in the same way. Other examples of clustering are grouping documents according to the topic.

The clustering technique can be widely used in various tasks. Some most common uses of this technique are:

- Market Segmentation
- Statistical data analysis
- Social network analysis
- Image segmentation
- Anomaly detection, etc.

Apart from these general usages, it is used by the Amazon in its recommendation system to provide the recommendations as per the past search of products. Netflix also uses this technique to recommend the movies and web-series to its users as per the watch history.

The below diagram explains the working of the clustering algorithm. We can see the different fruits are divided into several groups with similar properties.



Types of Clustering Methods

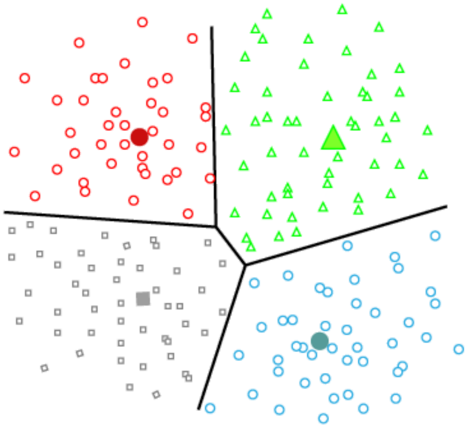
The clustering methods are broadly divided into Hard clustering (datapoint belongs to only one group) and Soft Clustering (data points can belong to another group also). But there are also other various approaches of Clustering exist. Below are the main clustering methods used in Machine learning:

1. Partitioning Clustering
2. Density-Based Clustering
3. Distribution Model-Based Clustering
4. Hierarchical Clustering
5. Fuzzy Clustering

Partitioning Clustering

It is a type of clustering that divides the data into non-hierarchical groups. It is also known as the centroid-based method. The most common example of partitioning clustering is the K-Means Clustering algorithm.

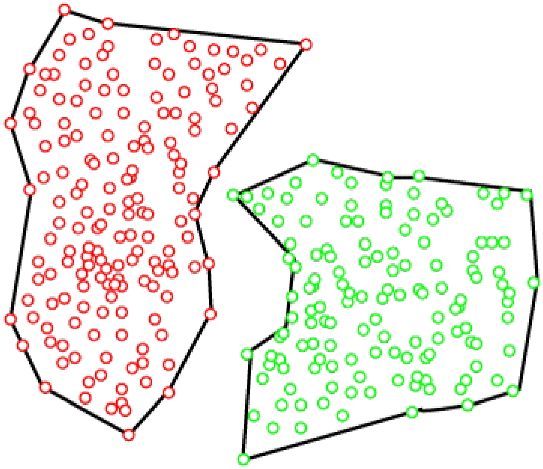
In this type, the dataset is divided into a set of k groups, where K is used to define the number of pre-defined groups. The cluster center is created in such a way that the distance between the data points of one cluster is minimum as compared to another cluster centroid.



Density-Based Clustering

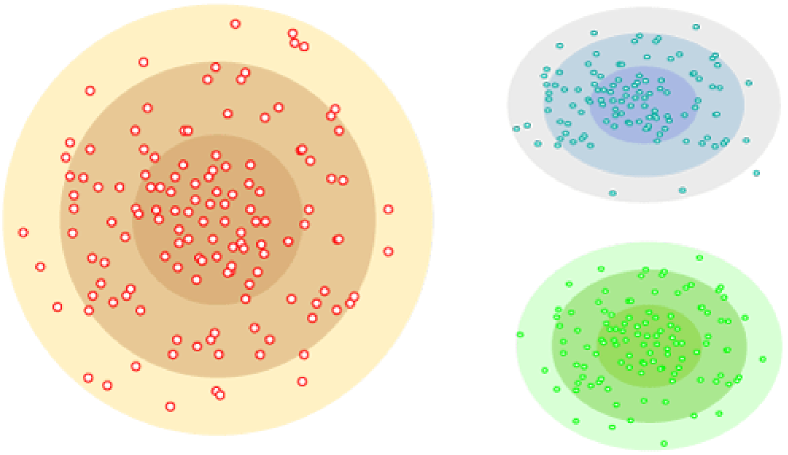
The density-based clustering method connects the highly-dense areas into clusters, and the arbitrarily shaped distributions are formed as long as the dense region can be connected. This algorithm does it by identifying different clusters in the dataset and connects the areas of high densities into clusters. The dense areas in data space are divided from each other by sparser areas.

These algorithms can face difficulty in clustering the data points if the dataset has varying densities and high dimensions.



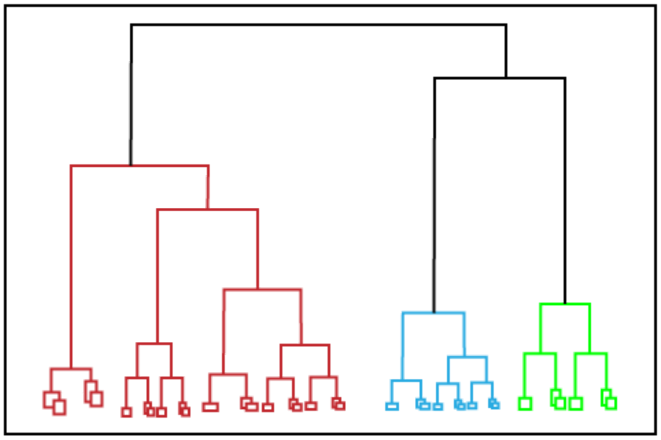
Distribution Model-Based Clustering

In the distribution model-based clustering method, the data is divided based on the probability of how a dataset belongs to a particular distribution. The grouping is done by assuming some distributions commonly Gaussian Distribution.
The example of this type is the Expectation-Maximization Clustering algorithm that uses Gaussian Mixture Models (GMM).



Hierarchical Clustering

Hierarchical clustering can be used as an alternative for the partitioned clustering as there is no requirement of pre-specifying the number of clusters to be created. In this technique, the dataset is divided into clusters to create a tree-like structure, which is also called a dendrogram. The observations or any number of clusters can be selected by cutting the tree at the correct level. The most common example of this method is the Agglomerative Hierarchical algorithm.



Fuzzy Clustering

Fuzzy clustering is a type of soft method in which a data object may belong to more than one group or cluster. Each dataset has a set of membership coefficients, which depend on the degree of membership to be in a cluster. Fuzzy C-means algorithm is the example of this type of clustering; it is sometimes also known as the Fuzzy k-means algorithm.

Clustering Algorithms

The Clustering algorithms can be divided based on their models that are explained above. There are different types of clustering algorithms published, but only a few are commonly used. The clustering algorithm is based on the kind of data that we are using. Such as, some algorithms need to guess the number of clusters in the given dataset, whereas some are required to find the minimum distance between the observation of the dataset.

Here we are discussing mainly popular Clustering algorithms that are widely used in machine learning:

1. K-Means algorithm: The k-means algorithm is one of the most popular clustering algorithms. It classifies the dataset by dividing the samples into different clusters of equal variances. The number of clusters must be specified in this algorithm. It is fast with fewer computations required, with the linear complexity of O(n).
2. Mean-shift algorithm: Mean-shift algorithm tries to find the dense areas in the smooth density of data points. It is an example of a centroid-based model, that works on updating the candidates for centroid to be the center of the points within a given region.

3. DBSCAN Algorithm: It stands for Density-Based Spatial Clustering of Applications with Noise. It is an example of a density-based model similar to the mean-shift, but with some remarkable advantages. In this algorithm, the areas of high density are separated by the areas of low density. Because of this, the clusters can be found in any arbitrary shape.
4. Expectation-Maximization Clustering using GMM: This algorithm can be used as an alternative for the k-means algorithm or for those cases where K-means can be failed. In GMM, it is assumed that the data points are Gaussian distributed.
5. Agglomerative Hierarchical algorithm: The Agglomerative hierarchical algorithm performs the bottom-up hierarchical clustering. In this, each data point is treated as a single cluster at the outset and then successively merged. The cluster hierarchy can be represented as a tree-structure.
6. Affinity Propagation: It is different from other clustering algorithms as it does not require to specify the number of clusters. In this, each data point sends a message between the pair of data points until convergence. It has $O(N^2T)$ time complexity, which is the main drawback of this algorithm.

Applications of Clustering

Below are some commonly known applications of clustering technique in Machine Learning:

- In Identification of Cancer Cells: The clustering algorithms are widely used for the identification of cancerous cells. It divides the cancerous and non-cancerous data sets into different groups.
- In Search Engines: Search engines also work on the clustering technique. The search result appears based on the closest object to the search query. It does it by grouping similar data objects in one group that is far from the other dissimilar objects. The accurate result of a query depends on the quality of the clustering algorithm used.
- Customer Segmentation: It is used in market research to segment the customers based on their choice and preferences.
- In Biology: It is used in the biology stream to classify different species of plants and animals using the image recognition technique.
- In Land Use: The clustering technique is used in identifying the area of similar lands use in the GIS database. This can be very useful to find that for what purpose the particular land should be used, that means for which purpose it is more suitable.
- Difference between Classification and Clustering

Classification	Clustering

Classification is a supervised learning approach where a specific label is provided to the machine to classify new observations. Here the machine needs proper testing and training for the label verification.	Clustering is an unsupervised learning approach where grouping is done on similarities basis.
Supervised learning approach.	Unsupervised learning approach.
It uses a training dataset.	It does not use a training dataset.
It uses algorithms to categorize the new data as per the observations of the training set.	It uses statistical concepts in which the data set is divided into subsets with the same features.
In classification, there are labels for training data.	In clustering, there are no labels for training data.
Its objective is to find which class a new object belongs to form the set of predefined classes.	Its objective is to group a set of objects to find whether there is any relationship between them.

- What is K-Means Algorithm?

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.

It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.

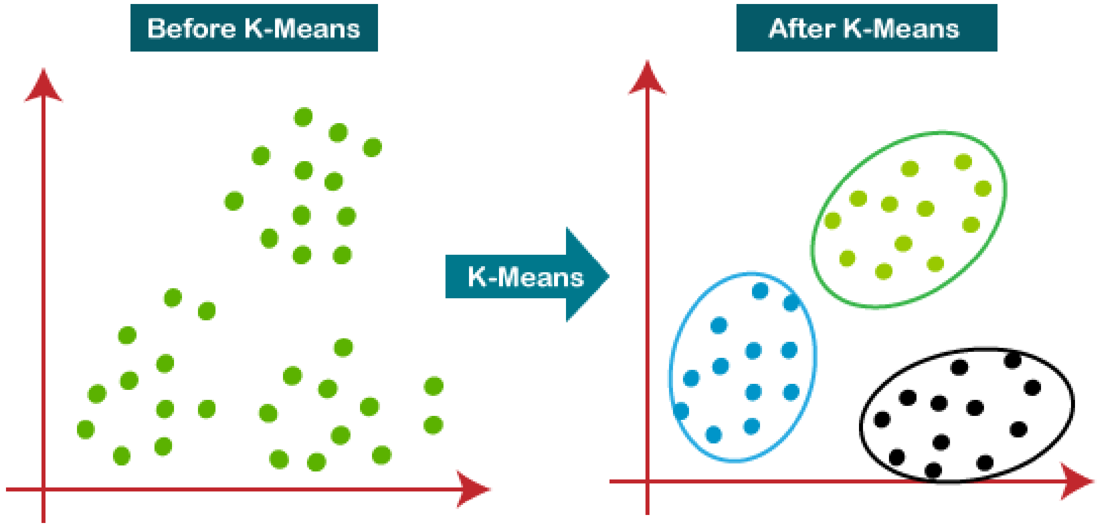
It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

The k-means clustering algorithm mainly performs two tasks:

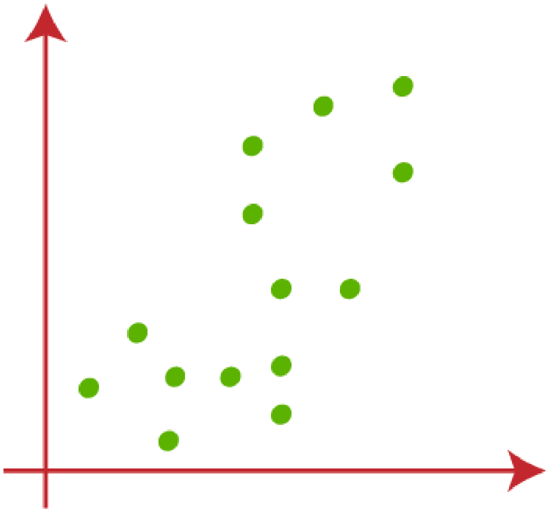
- Determines the best value for K center points or centroids by an iterative process.
- Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

Hence each cluster has datapoints with some commonalities, and it is away from other clusters.
The below diagram explains the working of the K-means Clustering Algorithm:

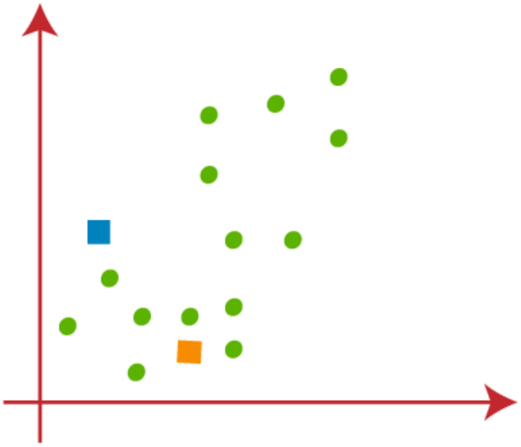


How does the K-Means Algorithm Work?

The working of the K-Means algorithm is explained in the below steps:
Step-1: Select the number K to decide the number of clusters.
Step-2: Select random K points or centroids. (It can be other from the input dataset).
Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters.
Step-4: Calculate the variance and place a new centroid of each cluster.
Step-5: Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.
Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.
Step-7: The model is ready.
Let's understand the above steps by considering the visual plots:
Suppose we have two variables M1 and M2. The x-y axis scatter plot of these two variables is given below:

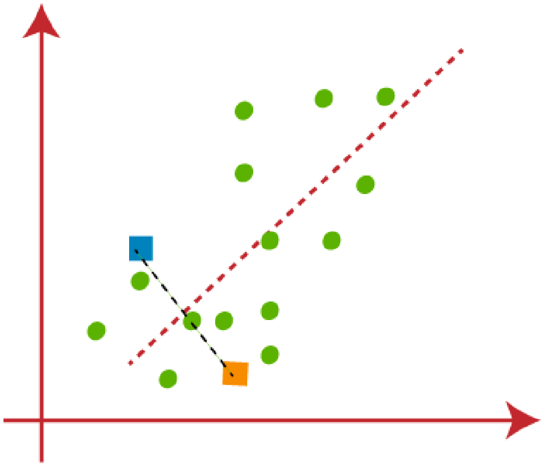


- Let's take number k of clusters, i.e., K=2, to identify the dataset and to put them into different clusters. It means here we will try to group these datasets into two different clusters.
- We need to choose some random k points or centroid to form the cluster. These points can be either the points from the dataset or any other point. So, here we are selecting the below two points as k points, which are not the part of our dataset. Consider the below image:

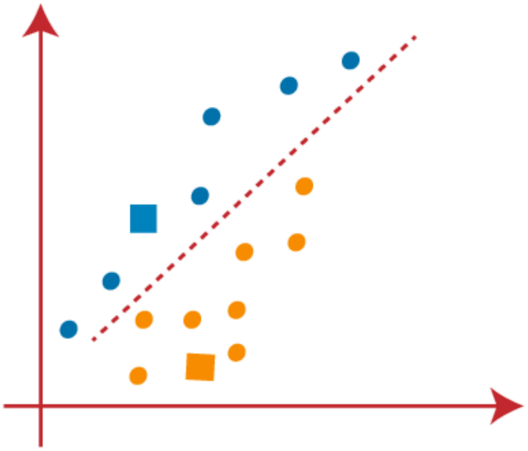


- Now we will assign each data point of the scatter plot to its closest K-point or centroid. We will compute it by applying some mathematics that we have studied to calculate the distance between two points. So, we will draw a median between both the centroids.

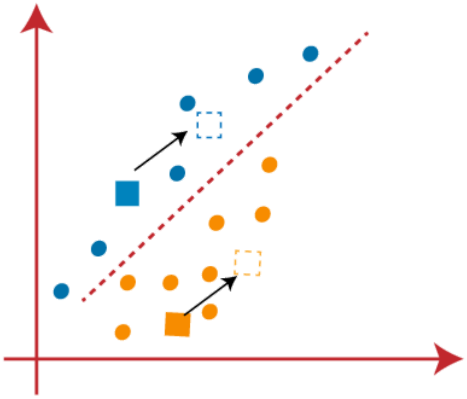
Consider the below image:



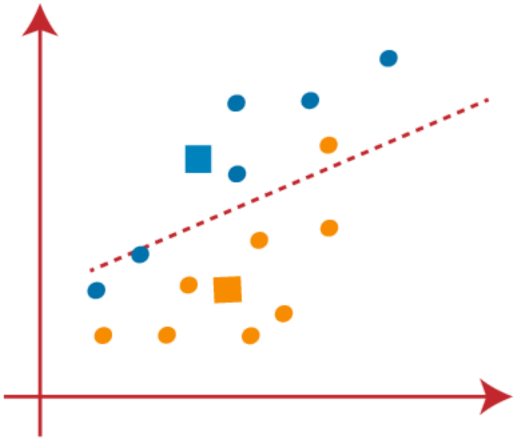
From the above image, it is clear that points left side of the line is near to the K1 or blue centroid, and points to the right of the line are close to the yellow centroid. Let's color them as blue and yellow for clear visualization.



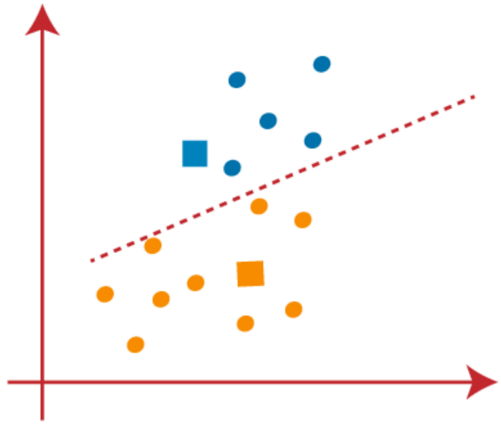
- ○ As we need to find the closest cluster, so we will repeat the process by choosing a new centroid. To choose the new centroids, we will compute the center of gravity of these centroids, and will find new centroids as below:



- Next, we will reassign each datapoint to the new centroid. For this, we will repeat the same process of finding a median line. The median will be like below image:

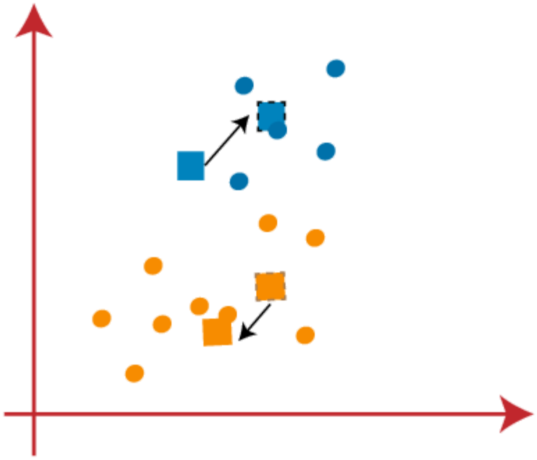


From the above image, we can see, one yellow point is on the left side of the line, and two blue points are right to the line. So, these three points will be assigned to new centroids.



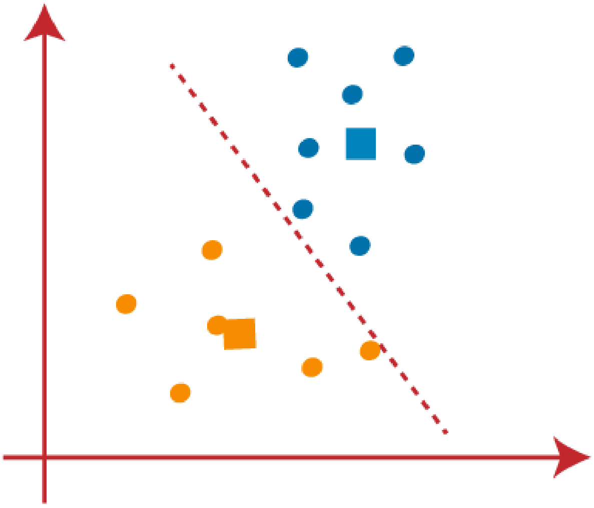
As reassignment has taken place, so we will again go to the step-4, which is finding new centroids or K-points.

- We will repeat the process by finding the center of gravity of centroids, so the new centroids will be as shown in the below image:

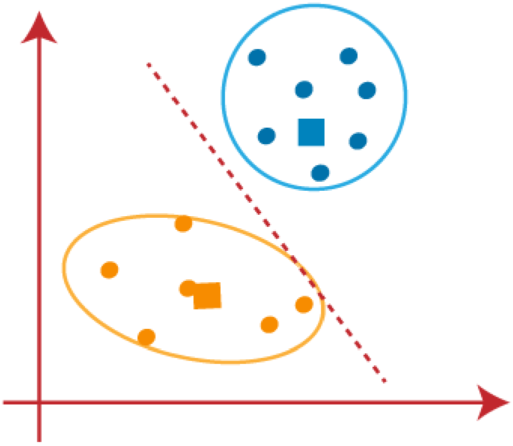


- As we got the new centroids so again will draw the median line and reassign the data points. So, the image will be:

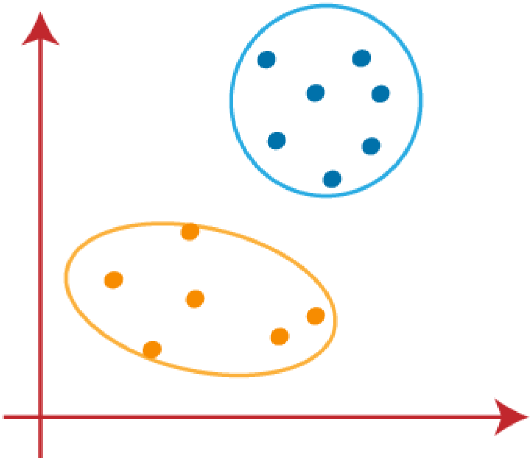
○



○ We can see in the above image; there are no dissimilar data points on either side of the line, which means our model is formed. Consider the below image:



As our model is ready, so we can now remove the assumed centroids, and the two final clusters will be as shown in the below image:



- K-means Clustering Numerical Example with Solution

Now that we have discussed the algorithm, let us solve a numerical problem on k means clustering. The problem is as follows.You are given 15 points in the Cartesian coordinate system as follows.

Point	Coordinates
A1	(2,10)
A2	(2,6)
A3	(11,11)
A4	(6,9)
A5	(6,4)
A6	(1,2)
A7	(5,10)
A8	(4,9)
A9	(10,12)
A10	(7,5)
A11	(9,11)

A12	(4,6)
A13	(3,10)
A14	(3,8)
A15	(6,11)

Input Dataset

We are also given the information that we need to make 3 clusters. It means we are given K=3.We will solve this numerical on k-means clustering using the approach discussed below. First, we will randomly choose 3 centroids from the given data. Let us consider A2 (2,6), A7 (5,10), and A15 (6,11) as the centroids of the initial clusters. Hence, we will consider that

- Centroid 1=(2,6) is associated with cluster 1.
- Centroid 2=(5,10) is associated with cluster 2.
- Centroid 3=(6,11) is associated with cluster 3.

Now we will find the euclidean distance between each point and the centroids. Based on the minimum distance of each point from the centroids, we will assign the points to a cluster. I have tabulated the distance of the given points from the clusters in the following table

Point	Distance from Centroid 1 (2,6)	Distance from Centroid 2 (5,10)	Distance from Centroid 3 (6,11)	Assigned Cluster
A1 (2,10)	4	3	4.123106	Cluster 2
A2 (2,6)	0	5	6.403124	Cluster 1
A3 (11,11)	10.29563	6.082763	5	Cluster 3
A4 (6,9)	5	1.414214	2	Cluster 2
A5 (6,4)	4.472136	6.082763	7	Cluster 1
A6 (1,2)	4.123106	8.944272	10.29563	Cluster 1
A7 (5,10)	5	0	1.414214	Cluster 2
A8 (4,9)	3.605551	1.414214	2.828427	Cluster 2
A9 (10,12)	10	5.385165	4.123106	Cluster 3

A10 (7,5)	5.09902	5.385165	6.082763	Cluster 1
A11 (9,11)	8.602325	4.123106	3	Cluster 3
A12 (4,6)	2	4.123106	5.385165	Cluster 1
A13 (3,10)	4.123106	2	3.162278	Cluster 2
A14 (3,8)	2.236068	2.828427	4.242641	Cluster 1
A15 (6,11)	6.403124	1.414214	0	Cluster 3

Results from 1st iteration of K means clustering

At this point, we have completed the first iteration of the k-means clustering algorithm and assigned each point into a cluster.

In the above table, you can observe that the point that is closest to the centroid of a given cluster is assigned to the cluster.

Now, we will calculate the new centroid for each cluster.

- In cluster 1, we have 6 points i.e. A2 (2,6), A5 (6,4), A6 (1,2), A10 (7,5), A12 (4,6), A14 (3,8). To calculate the new centroid for cluster 1, we will find the mean of the x and y coordinates of each point in the cluster. Hence, the new centroid for cluster 1 is (3.833, 5.167).
- In cluster 2, we have 5 points i.e. A1 (2,10), A4 (6,9), A7 (5,10) , A8 (4,9), and A13 (3,10). Hence, the new centroid for cluster 2 is (4, 9.6)
- In cluster 3, we have 4 points i.e. A3 (11,11), A9 (10,12), A11 (9,11), and A15 (6,11). Hence, the new centroid for cluster 3 is (9, 11.25).

Now that we have calculated new centroids for each cluster, we will calculate the distance of each data point from the new centroids. Then, we will assign the points to clusters based on their distance from the centroids. The results for this process have been given in the following table.

Point	Distance from Centroid 1 (3.833, 5.167)	Distance from centroid 2 (4, 9.6)	Distance from centroid 3 (9, 11.25)	Assigned Cluster
A1 (2,10)	5.169	2.040	7.111	Cluster 2

A2 (2,6)	2.013	4.118	8.750	Cluster 1
A3 (11,11)	9.241	7.139	2.016	Cluster 3
A4 (6,9)	4.403	2.088	3.750	Cluster 2
A5 (6,4)	2.461	5.946	7.846	Cluster 1
A6 (1,2)	4.249	8.171	12.230	Cluster 1
A7 (5,10)	4.972	1.077	4.191	Cluster 2
A8 (4,9)	3.837	0.600	5.483	Cluster 2
A9 (10,12)	9.204	6.462	1.250	Cluster 3
A10 (7,5)	3.171	5.492	6.562	Cluster 1
A11 (9,11)	7.792	5.192	0.250	Cluster 3
A12 (4,6)	0.850	3.600	7.250	Cluster 1
A13 (3,10)	4.904	1.077	6.129	Cluster 2
A14 (3,8)	2.953	1.887	6.824	Cluster 2
A15 (6,11)	6.223	2.441	3.010	Cluster 2

Results from 2nd iteration of K means clustering

Now, we have completed the second iteration of the k-means clustering algorithm and assigned each point into an updated cluster. In the above table, you can observe that the point closest to the new centroid of a given cluster is assigned to the cluster.

Now, we will calculate the new centroid for each cluster for the third iteration.

- In cluster 1, we have 5 points i.e. A2 (2,6), A5 (6,4), A6 (1,2), A10 (7,5), and A12 (4,6). To calculate the new centroid for cluster 1, we will find the mean of the x and y coordinates of each point in the cluster. Hence, the new centroid for cluster 1 is (4, 4.6).
- In cluster 2, we have 7 points i.e. A1 (2,10), A4 (6,9), A7 (5,10) , A8 (4,9), A13 (3,10), A14 (3,8), and A15 (6,11). Hence, the new centroid for cluster 2 is (4.143, 9.571)
- In cluster 3, we have 3 points i.e. A3 (11,11), A9 (10,12), and A11 (9,11). Hence, the new centroid for cluster 3 is (10, 11.333).

At this point, we have calculated new centroids for each cluster. Now, we will calculate the distance of each data point from the new centroids. Then, we will assign the points to clusters based on their distance from the centroids. The results for this process have been given in the following table.

Point	Distance from Centroid 1 (4, 4.6)	Distance from centroid 2 (4.143, 9.571)	Distance from centroid 3 (10, 11.333)	Assigned Cluster
A1 (2,10)	5.758	2.186	8.110	Cluster 2
A2 (2,6)	2.441	4.165	9.615	Cluster 1
A3 (11,11)	9.485	7.004	1.054	Cluster 3
A4 (6,9)	4.833	1.943	4.631	Cluster 2
A5 (6,4)	2.088	5.872	8.353	Cluster 1
A6 (1,2)	3.970	8.197	12.966	Cluster 1
A7 (5,10)	5.492	0.958	5.175	Cluster 2
A8 (4,9)	4.400	0.589	6.438	Cluster 2
A9 (10,12)	9.527	6.341	0.667	Cluster 3
A10 (7,5)	3.027	5.390	7.008	Cluster 1
A11 (9,11)	8.122	5.063	1.054	Cluster 3

A12 (4,6)	1.400	3.574	8.028	Cluster 1
A13 (3,10)	5.492	1.221	7.126	Cluster 2
A14 (3,8)	3.544	1.943	7.753	Cluster 2
A15 (6,11)	6.705	2.343	4.014	Cluster 2

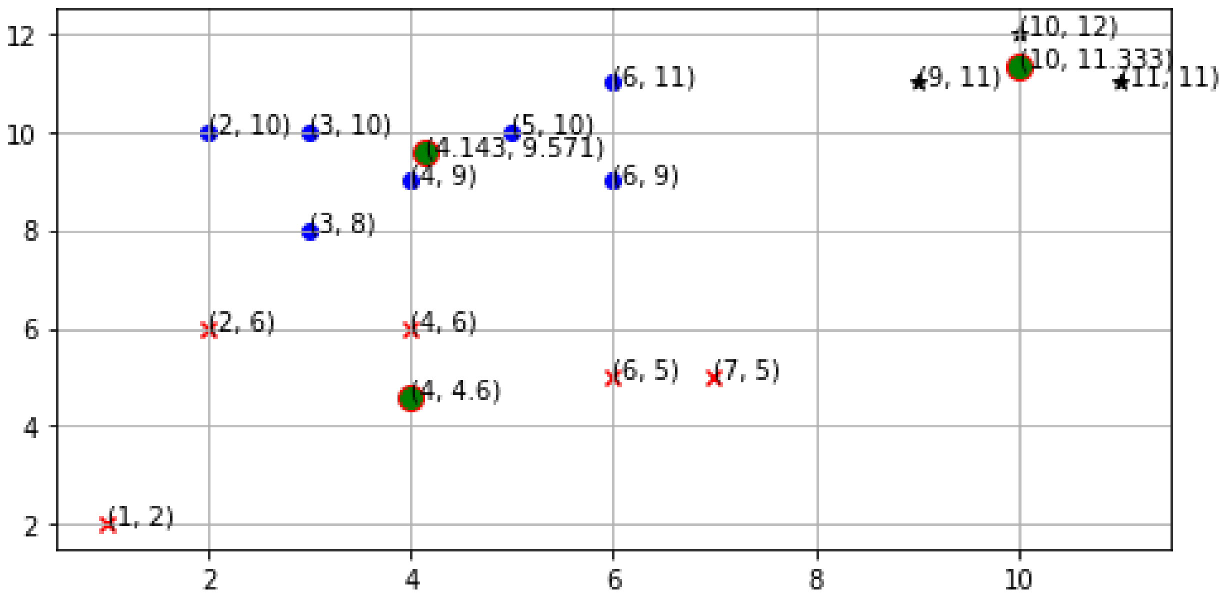
Results from 3rd iteration of K means clustering

Now, we have completed the third iteration of the k-means clustering algorithm and assigned each point into an updated cluster. In the above table, you can observe that the point that is closest to the new centroid of a given cluster is assigned to the cluster.

Now, we will calculate the new centroid for each cluster for the third iteration.

- In cluster 1, we have 5 points i.e. A2 (2,6), A5 (6,4), A6 (1,2), A10 (7,5), and A12 (4,6). To calculate the new centroid for cluster 1, we will find the mean of the x and y coordinates of each point in the cluster. Hence, the new centroid for cluster 1 is (4, 4.6).
- In cluster 2, we have 7 points i.e. A1 (2,10), A4 (6,9), A7 (5,10) , A8 (4,9), A13 (3,10), A14 (3,8), and A15 (6,11). Hence, the new centroid for cluster 2 is (4.143, 9.571)
- In cluster 3, we have 3 points i.e. A3 (11,11), A9 (10,12), and A11 (9,11). Hence, the new centroid for cluster 3 is (10, 11.333).

Here, you can observe that no point has changed its cluster compared to the previous iteration. Due to this, the centroid also remains constant. Therefore, we will say that the clusters have been stabilized. Hence, the clusters obtained after the third iteration are the final clusters made from the given dataset. If we plot the clusters on a graph, the graph looks like as follows.



Plot for K-Means Clustering

In the above plot, points in the clusters have been plotted using red, blue, and black markers. The centroids of the clusters have been marked using green circles.

Summary

Supervised Learning

Definition: Supervised learning is a type of machine learning where the model is trained on a labeled dataset. This means that for each input, the output (or label) is provided, and the model learns to map inputs to outputs by identifying patterns.

Popular Algorithms:

1. Naive Bayes Classifier
2. Decision Trees
3. Support Vector Machines (SVM)
4. K-Nearest Neighbors (KNN)
5. Linear Regression
6. Logistic Regression

Probabilistic Classifier

1. Bayesian Learning: This approach uses probabilities for predictions. The core concept involves using Bayes' theorem to update the probability estimate as more evidence or information becomes available.

2. Naive Bayes Classifier: Assumes conditional independence between features. It is a simple and effective algorithm often used for text classification tasks such as sentiment analysis.
3. Add-One Smoothing (Laplace Smoothing): This technique is used to handle zero probabilities in Naive Bayes by adding one to the count of each feature-label combination.

Decision Tree Learning

1. Entropy and Information Gain:
   - Entropy: Measures the impurity of a collection of training examples.
   - Information Gain: Measures the effectiveness of an attribute in classifying data.
2. Information Gain Equation:
   Information Gain=Entropy of parent−Weighted entropy of children

ID3 Algorithm: This algorithm creates a decision tree by using information gain to select the attribute that best classifies the data. It continues this process recursively for each node until it reaches a pure classification or certain criteria are met.

3. Overfitting and Pruning: Overfitting occurs when a model is too complex, capturing noise rather than the underlying pattern. Reduced error pruning is one way to avoid this by trimming branches of the tree that do not improve performance.
4. Discretizing Continuous Attributes: Converts continuous data into discrete intervals based on information gain, helping to simplify data for certain model.

Unsupervised Learning

Definition: Unsupervised learning is a machine learning approach where the model is trained on unlabeled data. The algorithm tries to understand the structure of the data without predefined labels.

Applications of Clustering:

1. Customer segmentation in marketing.
2. Image segmentation in computer vision.
3. Document clustering in natural language processing.

Difference between Clustering and Classification:

- Clustering: Groups similar data points together without any labels.
- Classification: Predicts predefined labels for data points.

K-Means Clustering Algorithm

1. K-means: It partitions data into KKK clusters based on the distance of data points from the cluster centroids. It is an iterative algorithm that updates the centroids until they stabilize.
2. Use Cases:

- Grouping similar items or patterns (e.g., customer segmentation).
- Identifying anomalies by finding outliers in clustered data.

Sample Questions

1. Multiple Choice Questions (MCQs)

1. Which algorithm is NOT a supervised learning algorithm?

a) K-means clustering
b) Logistic Regression
c) Decision Trees
d) K-Nearest Neighbors

2. What is the primary use of Naive Bayes classifiers?

a) Regression
b) Dimensionality Reduction
c) Classification
d) Clustering

3. What does the term "conditional independence" mean in Naive Bayes?

a) Each feature depends on every other feature.
b) Each feature is independent given the class label.
c) All features are independent.
d) None of the above

4. Entropy in information theory is used to measure:

a) Impurity
b) Prediction accuracy
c) Model performance
d) Dataset size

5. Which algorithm is best suited for grouping data without labels?

a) Naive Bayes
b) Decision Trees
c) K-Means
d) Support Vector Machine

6. In supervised learning, the model is trained on:
a) Labeled data
b) Unlabeled data
c) Random data
d) Outlier data
7. What type of problem is solved by logistic regression?
a) Classification
b) Clustering

c) Dimensionality reduction
d) Regression
8. Which algorithm is used to group similar data points in an unsupervised way?
a) Linear Regression
b) K-Means
c) Naive Bayes
d) Decision Trees
9 Entropy measures:
a) The distance between clusters
b) The accuracy of the model
c) The impurity within a dataset
d) The variance of attributes
10. The Naive Bayes algorithm is based on:
a) The law of large numbers
b) Bayes' theorem
c) Gradient descent
d) Euclidean distance
11. Which term describes the problem of a model learning noise rather than signal?
a) Underfitting
b) Overfitting
c) Generalization
d) Optimization
12. Information gain helps to:
a) Increase model size
b) Measure the effectiveness of an attribute
c) Decrease prediction accuracy
d) Predict the class label
13. In clustering, which algorithm is primarily used to find centroids?
a) Decision Tree
b) K-Means
c) Naive Bayes
d) SVM
14. The ID3 algorithm splits data based on:
a) Random selection
b) Gini Index
c) Information gain
d) Cost function
15. Conditional independence in Naive Bayes refers to:
a) All features being independent of each other
b) Features being independent given the class

c) Classes being dependent on the feature
d) Features being dependent on each other
16. Which of the following algorithms can handle both numerical and categorical data?
a) Naive Bayes
b) K-Means
c) Decision Trees
d) Linear Regression
17. Which learning algorithm requires labeled data to function?
a) K-Means
b) Clustering
c) Logistic Regression
d) Dimensionality reduction
18. The goal of clustering is to:
a) Classify data points
b) Reduce data dimensionality
c) Group similar data points together
d) Label new data points
19. What is the purpose of add-one smoothing in Naive Bayes?
a) To handle missing values
b) To deal with zero probabilities
c) To calculate entropy
d) To split continuous attributes
20. A high entropy value in a dataset means:
a) Low purity of examples
b) High purity of examples
c) High prediction accuracy
d) Low feature importance
21. Which of the following is an example of a classification problem?
a) Market segmentation
b) Predicting stock prices
c) Spam email detection
d) Clustering customer groups
22. In supervised learning, underfitting occurs when:
a) The model learns noise in the data
b) The model generalizes well to unseen data
c) The model fails to capture the patterns in data
d) The model perfectly predicts training examples
23. Which of the following is NOT a feature of Naive Bayes?
a) Assumes conditional independence of features
b) Suitable for high-dimensional data

c) Requires labeled data
d) Sensitive to class imbalance
24. The purpose of reduced error pruning in decision trees is to:
a) Improve accuracy on test data
b) Increase tree size
c) Enhance entropy
d) Select attributes randomly
25. K-means clustering is best suited for:
a) Finding correlations
b) Segmenting similar groups
c) Predicting continuous values
d) Text classification
26. Supervised learning primarily focuses on:
a) Pattern discovery
b) Label prediction
c) Data generation
d) Data compression
27. Which metric is typically used to evaluate the purity of a node in a decision tree?
a) Entropy
b) Euclidean distance
c) Confusion matrix
d) R-squared
28. A decision tree's root node is chosen based on the attribute with the highest:
a) Variance
b) Information gain
c) Loss
d) Entropy
29. In Naive Bayes, the sum of all probabilities in a given class should equal:
a) 1
b) 0
c) 100
d) 50
30. Which of the following is used to prevent a model from overfitting?
a) Training longer
b) Increased data collection
c) Reduced error pruning
d) Adding noise to data
31. Which method is commonly used for dimensionality reduction in unsupervised learning?
a) PCA (Principal Component Analysis)
b) Naive Bayes

c) Decision Tree
d) K-Nearest Neighbors
32. Which term describes the act of grouping data without predefined labels?
a) Classification
b) Clustering
c) Regression
d) Association
33. In clustering, a centroid represents:
a) The most distant point in the dataset
b) The middle point in each cluster
c) The attribute with highest information gain
d) The decision boundary in classification
34. Which of the following is NOT a clustering algorithm?
a) K-means
b) Agglomerative Hierarchical Clustering
c) Naive Bayes
d) DBSCAN
35. The Naive Bayes classifier calculates probability based on:
a) Decision tree structure
b) Feature conditional independence
c) Gradient descent
d) Distance measures

2. Very Short Answer Questions (1-2 Sentences)

1. Define supervised learning.
2. What is a probabilistic classifier?
3. Why is Naive Bayes classifier considered “naive”?
4. Explain the concept of entropy.
5. What does the ID3 algorithm optimize for when building a decision tree?
6. What is the main drawback of using the Naive Bayes classifier?
7. Why is overfitting a concern in machine learning?
8. What is reduced error pruning?
9. Define clustering in the context of machine learning.
10. Give an example of an application of unsupervised learning.

3. Short Answer Questions (3 Marks)

1. Explain the concept of conditional independence with an example.
2. Describe the steps involved in the ID3 algorithm.
3. List three applications of the Naive Bayes classifier.

4. Explain why clustering is considered an unsupervised learning technique.
5. Describe add-one smoothing and its importance in Naive Bayes.
6. What is entropy and how does it relate to information gain?
7. Explain the difference between supervised and unsupervised learning.
8. Give a simple example of how K-means clustering works.
9. What is meant by discretizing continuous-valued attributes?
10. How does overfitting impact the performance of a machine learning model?

4. Long Answer Questions (5 Marks)

1. Explain the ID3 algorithm with an illustrative example.
2. Discuss the concept of information gain and its role in building decision trees.
3. Describe the Naive Bayes classifier, its assumptions, and applications.
4. Compare and contrast clustering and classification.
5. Explain the K-means clustering algorithm in detail, including its limitations.
6. Describe overfitting in detail and explain two methods to address it.
7. How does conditional independence simplify the computations in Naive Bayes?
8. Describe a real-world application of clustering and how it adds value.
9. Explain how entropy is calculated and used in decision tree learning.
10. Describe the process of reduced error pruning in decision trees.

Semester- IV Unit 5

Biological motivation for Artificial Neural Networks(ANN):

The study of artificial neural networks (ANNs) has been inspired in part by the observation that biological learning systems are built of very complex webs of interconnected neurons.

• Artificial neural networks are built out of a densely interconnected set of simple units, where each unit takes a number of real-valued inputs (possibly the outputs of other units) and produces a single real-valued output (which may become the input to many other units).

• The human brain is estimated to contain a densely interconnected network of approximately 10^{11} neurons, each connected, on average, to 10^4 others.

Neuron activity is typically inhibited through connections to other neurons.

Artificial neural networks (ANNs) are inspired by the biological structure of the human brain, which is made up of a complex network of interconnected neurons:

Structure

ANNs are made up of simple, densely interconnected units that process inputs and produce outputs.

Function

Neurons in the brain send electrical signals to each other to process information. In ANNs, neurons receive inputs, process them, and generate outputs.

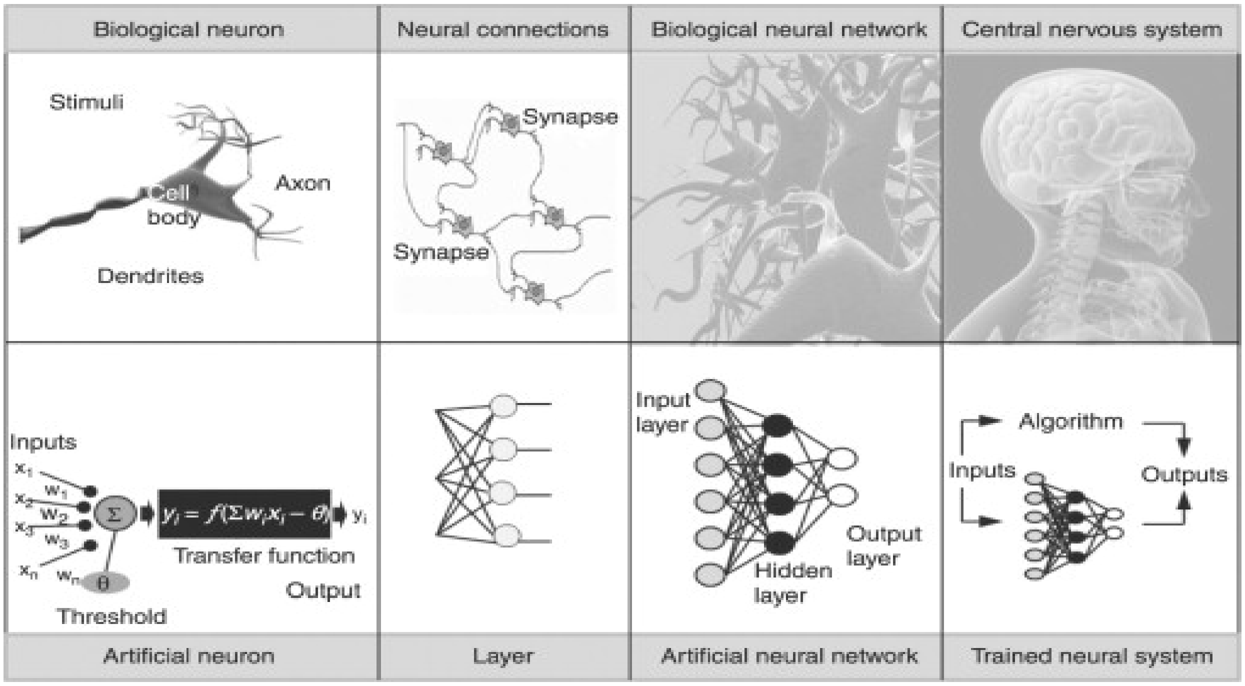
Learning

ANNs are known for their learning capability. In biological systems, adaptation is a process that occurs over multiple generations, where individuals slowly change to better fit their environment. In ANNs, learning is an operation that can be performed on a single individual.

Here are some other characteristics of ANNs:

- Activation function: Each neuron has an activation function, which is a mathematical equation that determines when the neuron generates an impulse.
- Connections: Each connection in the network has a weight associated with it.
- Threshold: Neurons calculate a weighted sum of inputs and compare it to a threshold.

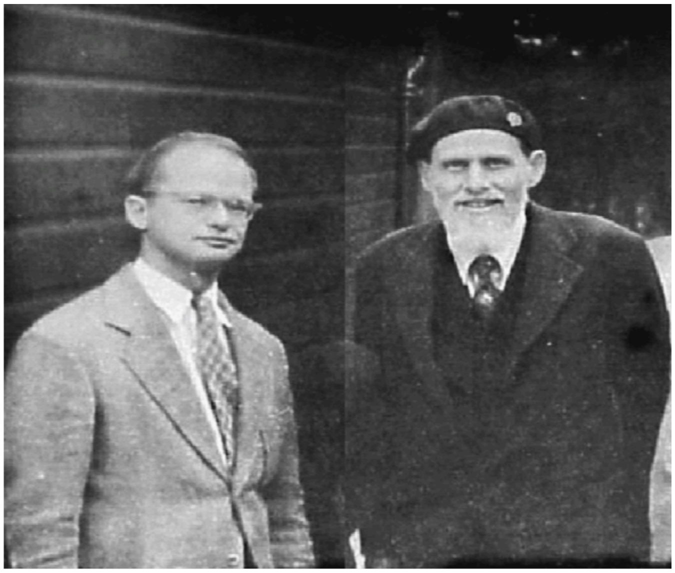
কৃত্রিম নিউরাল নেটওয়ার্ক (ANN) মানব মস্তিষ্কের জৈবিক গঠন দ্বারা অনুপ্রাণিত , যা আন্তঃসংযুক্ত নিউরনের একটি জটিল নেটওয়ার্ক দ্বারা গঠিত:



Biological Motivation:

- ANNs are inspired by the structure and function of biological neurons. The firing of a neuron upon reaching a threshold influenced McCulloch and Pitts (1943) to propose a simple mathematical model of a neuron.

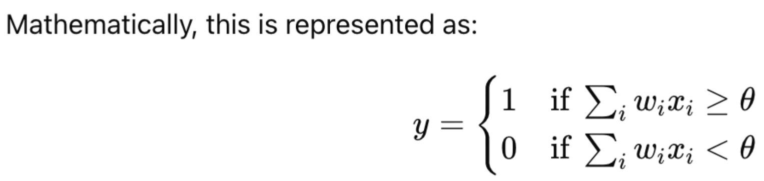
A simple mathematical model of a neuron (McCulloch and Pitts(1943))



The McCulloch and Pitts model, introduced in 1943, was one of the first mathematical models of a neuron, inspired by the biological workings of the human brain. This model presents a simple binary neuron that either "fires" (outputs 1) or remains inactive (outputs 0), based on a threshold mechanism.

Key Components of the McCulloch and Pitts Neuron Model

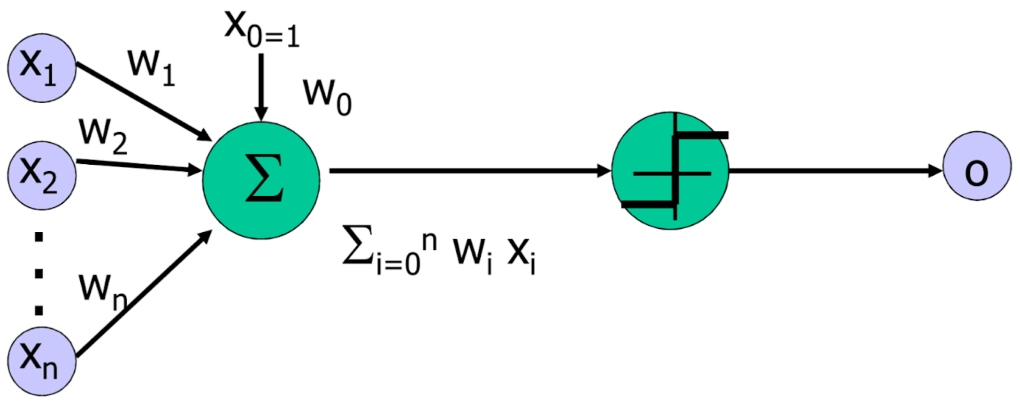
1. Inputs and Weights:
    - The neuron receives multiple inputs, each associated with a weight. These weights represent the strength or importance of each input in determining the output.
2. Weighted Sum:
    - Each input x_i is multiplied by its corresponding weight w_i.
    - The neuron calculates the weighted sum of all inputs: Weighted Sum=$\sum w_i x_i$.
3. Threshold Function:
    - The neuron has a threshold value, θ.
    - If the weighted sum of the inputs is greater than or equal to this threshold, the neuron "fires" and outputs 1.
    - If the weighted sum is less than the threshold, the neuron does not fire and outputs 0.



Mathematically, this is represented as:

$$y = \begin{cases} 1 & \text{if } \sum_i w_i x_i \geq \theta \\ 0 & \text{if } \sum_i w_i x_i < \theta \end{cases}$$

Perceptron



$$o(x_0,\ldots,x_n)= \begin{cases} 1 & \text{if } \Sigma_{i=0}^{n} w_i x_i > 0 \\ -1 & \text{otherwise} \end{cases}$$

Interpretation and Significance

- Binary Output: This binary output represents a simple on-off switch similar to neuron firing in the brain.
- Logical Operations: The McCulloch and Pitts model could perform basic logical functions like AND, OR, and NOT, which is significant in the foundation of neural network models.
- Limitations: Although groundbreaking, this model is limited to linearly separable problems and lacks the capacity for more complex, nonlinear decision boundaries. This limitation paved the way for more advanced models incorporating continuous activation functions and multi-layer architectures.

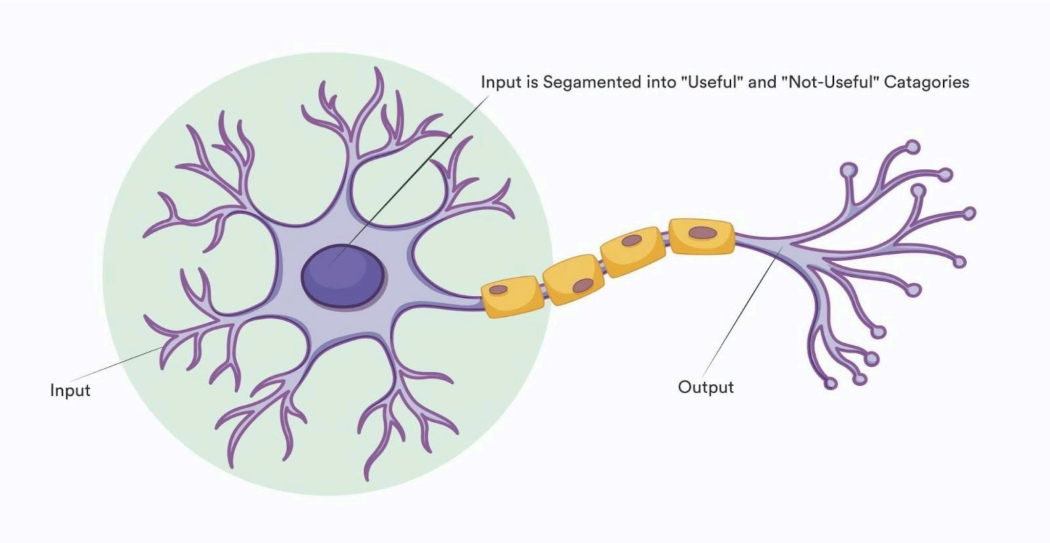
একটি পারসেপ্ট্রন হল একটি একক-স্তর নিউরাল নেটওয়ার্ক যা ইনপুটকে দুটি বিভাগে শ্রেণীবদ্ধ করে । এটি এক ধরনের লিনিয়ার ক্লাসিফায়ার যা ভবিষ্যদ্বাণী করার জন্য একটি বৈশিষ্ট্য ভেক্টরের সাথে ওজনের একটি সেটকে একত্রিত করতে একটি রৈখিক ভবিষ্যদ্বাণীকারী ফাংশন ব্যবহার করে।

এটা কি করে	ইনপুটগুলিকে দুটি বিভাগে শ্রেণীবদ্ধ করে

এটা কিভাবে কাজ করে	প্রশিক্ষণের ডেটার উপর ভিত্তি করে সিদ্ধান্তের সীমানার ওজন শিখে
এটা কি ফেরত	একটি একক বাইনারি আউটপুট, হয় 1 বা 0।
সুবিধা	বাস্তবায়ন, প্রশিক্ষণ, এবং বুঝতে সহজ
সীমাবদ্ধতা	প্রশিক্ষণের অসুবিধার কারণে গভীর শিক্ষায় সাধারণত ব্যবহৃত হয় না

Concept of activation function: threshold function and Sigmoid function

Definition: In artificial neural networks, each neuron forms a weighted sum of its inputs and passes the resulting scalar value through a function referred to as an activation function.”


Input is Segamented into "Useful" and "Not-Useful" Catagories
Input
Output

In humans, our brain receives input from the outside world, performs processing on the neuron receiving input and activates the neuron tail to generate required decisions. Similarly, in neural networks, we provide input as images, sounds, numbers, etc., and processing is performed on the artificial neuron, with an algorithm activating the correct final neuron layer to generate results.

Why do we need activation functions?

An activation function determines if a neuron should be activated or not activated. This implies

that it will use some simple mathematical operations to determine if the neuron's input to the network is relevant or not relevant in the prediction process.
The ability to introduce non-linearity to an artificial neural network and generate output from a collection of input values fed to a layer is the purpose of the activation function.

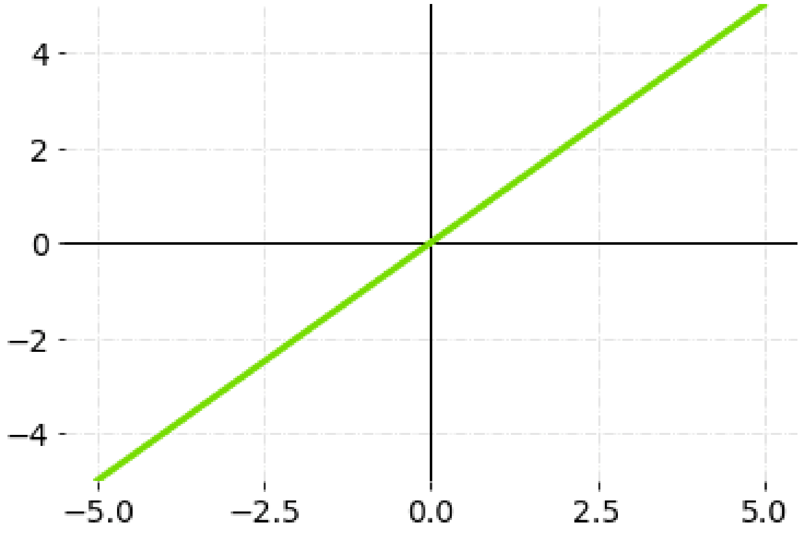
Types of Activation functions

Activation functions can be divided into three types:

1. Linear Activation Function
2. Binary Step Function
3. Non-linear Activation Functions

Linear Activation Function

The linear activation function, often called the identity activation function, is proportional to the input. The range of the linear activation function will be (-∞ to ∞). The linear activation function simply adds up the weighted total of the inputs and returns the result.



Linear Activation Function—Graph

Mathematically, it can be represented as:

$$f(x) = x$$

Linear Activation Function—Equation

Pros and Cons

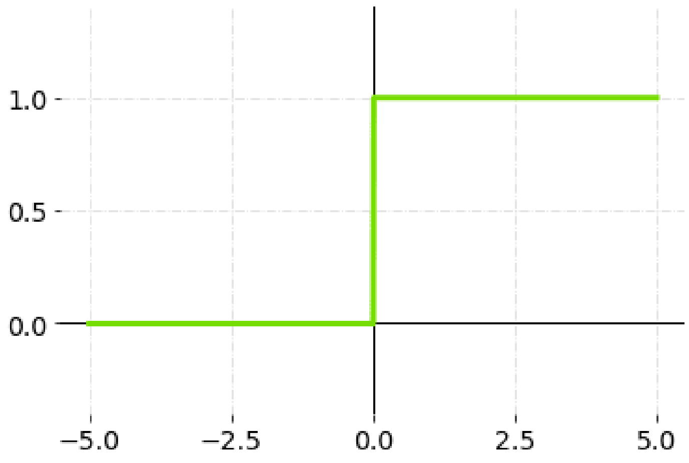
1. It is not a binary activation because the linear activation function only delivers a range of activations. We can surely connect a few neurons together, and if there are multiple activations, we can calculate the max (or soft max) based on that.

2. The derivative of this activation function is a constant. That is to say, the gradient is unrelated to the x (input).

Binary Step Activation Function

A threshold value determines whether a neuron should be activated or not activated in a binary step activation function.

The activation function compares the input value to a threshold value. If the input value is greater than the threshold value, the neuron is activated. It's disabled if the input value is less than the threshold value, which means its output isn't sent on to the next or hidden layer.



Binary Step Function—Graph

Mathematically, the binary activation function can be represented as:

$$f(x) = \begin{cases} 0 & for\ x < 0 \\ 1 & for\ x \geqslant 0 \end{cases}$$

Binary Step Activation Function—Equation

Pros and Cons

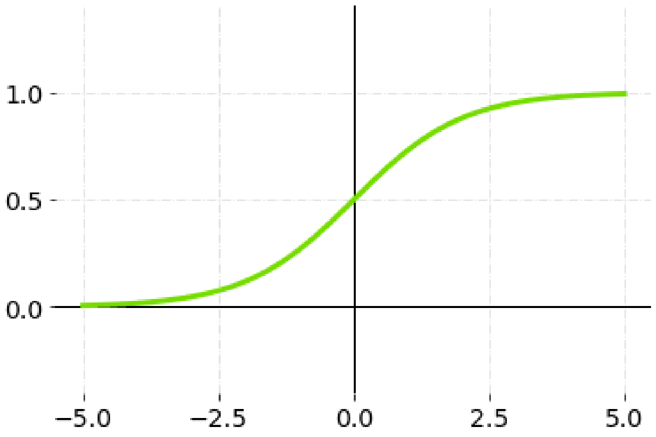
- It cannot provide multi-value outputs—for example, it cannot be used for multi-class classification problems.
- The step function's gradient is zero, which makes the back propagation procedure difficult.

Non-linear Activation Functions

The non-linear activation functions are the most-used activation functions. They make it uncomplicated for an artificial neural network model to adapt to a variety of data and to differentiate between the outputs.
Non-linear activation functions allow the stacking of multiple layers of neurons, as the output would now be a non-linear combination of input passed through multiple layers. Any output can be represented as a functional computation output in a neural network.
These activation functions are mainly divided on the basis of their range and curves. The remainder of this article will outline the major nonlinear activation functions used in neural networks.

Sigmoid

Sigmoid accepts a number as input and returns a number between 0 and 1. It's simple to use and has all the desirable qualities of activation functions: nonlinearity, continuous differentiation, monotonicity, and a set output range.
This is mainly used in binary classification problems. This sigmoid function gives the probability of an existence of a particular class.



Sigmoid Activation Function—Graph

Mathematically, it can be represented as:

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

Sigmoid Activation Function—Equation

Pros and Cons

1. It is non-linear in nature. Combinations of this function are also non-linear, and it will give an analogue activation, unlike the binary step activation function. It has a smooth gradient too, and It's good for a classifier type problem.
2. The output of the activation function is always going to be in the range (0,1) compared to (-∞, ∞) of the linear activation function. As a result, we've defined a range for our activations.
3. Sigmoid function gives rise to a problem of "Vanishing gradients" and Sigmoids saturate and kill gradients.
4. Its output isn't zero centered4, and it makes the gradient updates go too far in different directions. The output value is between zero and one, so it makes optimization harder.
5. The network either refuses to learn more or is extremely slow.

Feature	Threshold Function	Sigmoid Function
Output Range	0 or 1	0 to 1
Continuity	Discrete (not continuous)	Continuous
Differentiable	No	Yes
Suitability	Binary classification	Binary classification, probabilistic output
Common Usage	Perceptrons, single-layer	Multilayer neural networks, logistic regression

Practical Usage

In neural networks, the sigmoid function is often preferred over the threshold function for training with backpropagation, as it provides a gradient for error correction. The threshold function is useful for simpler models or when interpreting neural network outputs in strict binary terms is essential.

By using activation functions like sigmoid and others, neural networks gain the ability to approximate complex, nonlinear functions, enabling them to perform more sophisticated tasks.

Perceptron as a linear classifier, perceptron training rule

Basic Components of Perceptron

Perceptron is a type of artificial neural network, which is a fundamental concept in machine learning. The basic components of a perceptron are:

1. Input Layer: The input layer consists of one or more input neurons, which receive input signals from the external world or from other layers of the neural network.
2. Weights: Each input neuron is associated with a weight, which represents the strength of the connection between the input neuron and the output neuron.
3. Bias: A bias term is added to the input layer to provide the perceptron with additional flexibility in modeling complex patterns in the input data.
4. Activation Function: The activation function determines the output of the perceptron based on the weighted sum of the inputs and the bias term. Common activation functions used in perceptrons include the step function, sigmoid function, and ReLU function.
5. Output: The output of the perceptron is a single binary value, either 0 or 1, which indicates the class or category to which the input data belongs.

6. Training Algorithm: The perceptron is typically trained using a supervised learning algorithm such as the perceptron learning algorithm or backpropagation. During training, the weights and biases of the perceptron are adjusted to minimize the error between the predicted output and the true output for a given set of training examples.
7. Overall, the perceptron is a simple yet powerful algorithm that can be used to perform binary classification tasks and has paved the way for more complex neural networks used in deep learning today.

Types of Perceptron:

1. Single layer: Single layer perceptron can learn only linearly separable patterns.
2. Multilayer: Multilayer perceptrons can learn about two or more layers having a greater processing power.

The Perceptron algorithm learns the weights for the input signals in order to draw a linear decision boundary.

Note: Supervised Learning is a type of Machine Learning used to learn models from labeled training data. It enables output prediction for future or unseen data. Let us focus on the Perceptron Learning Rule in the next section.

Perceptron in Machine Learning

The most commonly used term in Artificial Intelligence and Machine Learning (AIML) is Perceptron. It is the beginning step of learning coding and Deep Learning technologies, which consists of input values, scores, thresholds, and weights implementing logic gates. Perceptron is the nurturing step of an Artificial Neural Link. In 19h century, Mr. Frank Rosenblatt invented the Perceptron to perform specific high-level calculations to detect input data capabilities or business intelligence. However, now it is used for various other purposes.

History of Perceptron

The perceptron was introduced by Frank Rosenblatt in 1958, as a type of artificial neural network capable of learning and performing binary classification tasks. Rosenblatt was a psychologist and computer scientist who was interested in developing a machine that could learn and recognize patterns in data, inspired by the workings of the human brain.

The perceptron was based on the concept of a simple computational unit, which takes one or more inputs and produces a single output, modeled after the structure and function of a neuron in the brain. The perceptron was designed to be able to learn from examples and adjust its parameters to improve its accuracy in classifying new examples.

The perceptron algorithm was initially used to solve simple problems, such as recognizing handwritten characters, but it soon faced criticism due to its limited capacity to learn complex patterns and its inability to handle non-linearly separable data. These limitations led to the decline of research on perceptrons in the 1960s and 1970s.

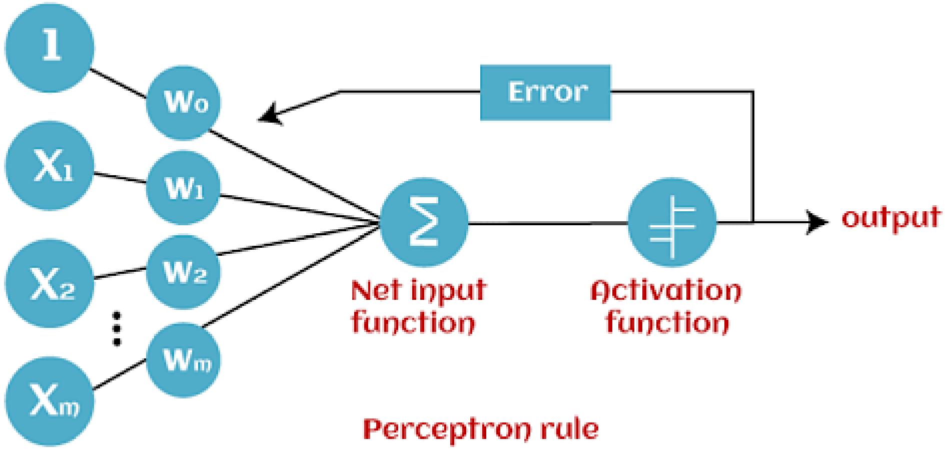
However, in the 1980s, the development of backpropagation, a powerful algorithm for training multi-layer neural networks, renewed interest in artificial neural networks and sparked a new era of research and innovation in machine learning. Today, perceptrons are regarded as the simplest form of artificial neural networks and are still widely used in applications such as image recognition, natural language processing, and speech recognition.

What is the Perceptron Model in Machine Learning?

A machine-based algorithm used for supervised learning of various binary sorting tasks is called Perceptron. Furthermore, Perceptron also has an essential role as an Artificial Neuron or Neural link in detecting certain input data computations in business intelligence. A perceptron model is also classified as one of the best and most specific types of Artificial Neural networks. Being a supervised learning algorithm of binary classifiers, we can also consider it a single-layer neural network with four main parameters: input values, weights and Bias, net sum, and an activation function.

How Does Perceptron Work?

AS discussed earlier, Perceptron is considered a single-layer neural link with four main parameters. The perceptron model begins with multiplying all input values and their weights, then adds these values to create the weighted sum. Further, this weighted sum is applied to the activation function 'f' to obtain the desired output. This activation function is also known as the step function and is represented by 'f.'



This step function or Activation function is vital in ensuring that output is mapped between (0,1) or (-1,1). Take note that the weight of input indicates a node's strength. Similarly, an input value gives the ability the shift the activation function curve up or down.

Step 1: Multiply all input values with corresponding weight values and then add to calculate the weighted sum. The following is the mathematical expression of it:

∑wi*xi = x1*w1 + x2*w2 + x3*w3+……..x4*w4

Add a term called bias ‘b’ to this weighted sum to improve the model’s performance.

Step 2: An activation function is applied with the above-mentioned weighted sum giving us an output either in binary form or a continuous value as follows:

We have already discussed the types of Perceptron models in the Introduction. Here, we shall give a more profound look at this:

1. Single Layer Perceptron model: One of the easiest ANN(Artificial Neural Networks) types consists of a feed-forward network and includes a threshold transfer inside the model. The main objective of the single-layer perceptron model is to analyze the linearly separable objects with binary outcomes. A Single-layer perceptron can learn only linearly separable patterns.
2. Multi-Layered Perceptron model: It is mainly similar to a single-layer perceptron model but has more hidden layers.

Forward Stage: From the input layer in the on stage, activation functions begin and terminate on the output layer.

Backward Stage: In the backward stage, weight and bias values are modified per the model’s requirement. The backstage removed the error between the actual output and demands originating backward on the output layer. A multilayer perceptron model has a greater processing power and can process linear and non-linear patterns. Further, it also implements logic gates such as AND, OR, XOR, XNOR, and NOR.

Advantages:

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

Disadvantages:

- In multi-layered perceptron model, computations are time-consuming and complex.
- It is tough to predict how much the dependent variable affects each independent variable.
- The model functioning depends on the quality of training.

Characteristics of the Perceptron Model

The following are the characteristics of a Perceptron Model:

1. It is a machine learning algorithm that uses supervised learning of binary classifiers.
2. In Perceptron, the weight coefficient is automatically learned.
3. Initially, weights are multiplied with input features, and then the decision is made whether the neuron is fired or not.

4. The activation function applies a step rule to check whether the function is more significant than zero.
5. The linear decision boundary is drawn, enabling the distinction between the two linearly separable classes +1 and -1.
6. If the added sum of all input values is more than the threshold value, it must have an output signal; otherwise, no output will be shown.

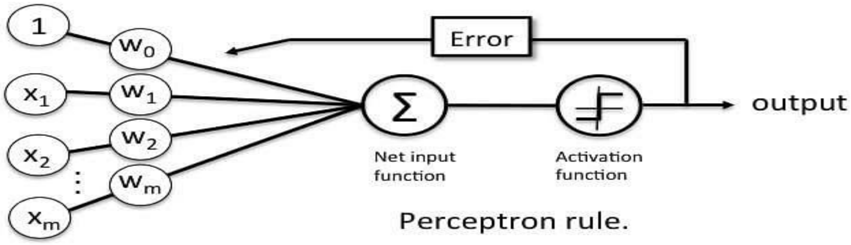
Limitation of Perceptron Model

The following are the limitation of a Perceptron model:

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

Perceptron Learning Rule

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



Perceptron rule.

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

Perceptron Function

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

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

In the equation given above:

- “w” = vector of real-valued weights

- “b” = bias (an element that adjusts the boundary away from origin without any dependence on the input value)
- “x” = vector of input x values

$$\sum_{i=1}^{m} w_i x_i$$

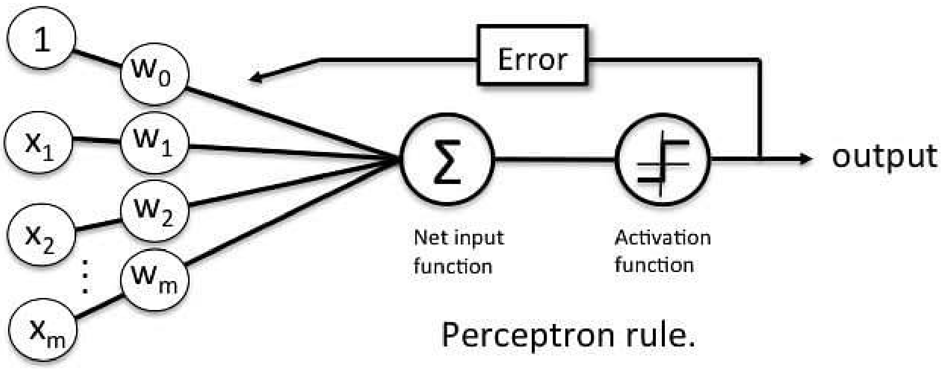
- “m” = number of inputs to the Perceptron

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

Let us learn the inputs of a perceptron in the next section.

Inputs of a Perceptron

A Perceptron accepts inputs, moderates them with certain weight values, then applies the transformation function to output the final result. The image below shows a Perceptron with a Boolean output.



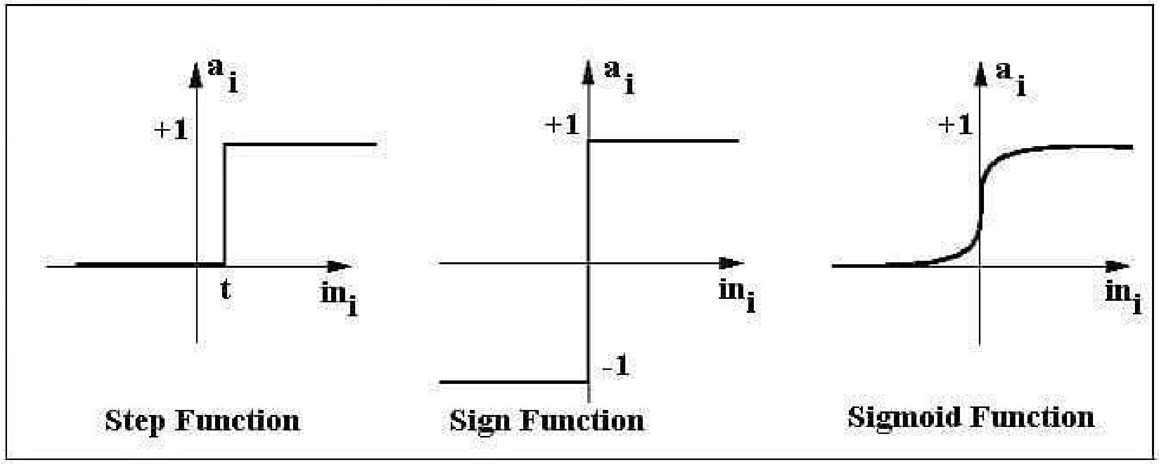
Perceptron rule.

A Boolean output is based on inputs such as salaried, married, age, past credit profile, etc. It has only two values: Yes and No or True and False. The summation function “∑” multiplies all inputs of “x” by weights “w” and then adds them up as follows:

$$w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n$$

Activation Functions of Perceptron

The activation function applies a step rule (convert the numerical output into +1 or -1) to check if the output of the weighting function is greater than zero or not.



For example:
If ∑ wixi> 0 => then final output “o” = 1 (issue bank loan)
Else, final output “o” = -1 (deny bank loan)
Step function gets triggered above a certain value of the neuron output; else it outputs zero. Sign Function outputs +1 or -1 depending on whether neuron output is greater than zero or not. Sigmoid is the S-curve and outputs a value between 0 and 1.

Output of Perceptron

Perceptron with a Boolean output:
Inputs: x1…xn
Output: o(x1….xn)

$$o(x_1, \ldots, x_n) = \begin{cases} 1 \text{ if } w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n > 0 \\ -1 \text{ otherwise} \end{cases}$$

Weights: wi=> contribution of input xi to the Perceptron output;
w0=> bias or threshold
If ∑w.x > 0, output is +1, else -1. The neuron gets triggered only when weighted input reaches a certain threshold value.

$$o(\vec{x}) = sgn(\vec{w} \cdot \vec{x})$$

$$sgn(y) = \begin{cases} 1 \text{ if } y > 0 \\ -1 \text{ otherwise} \end{cases}$$

An output of +1 specifies that the neuron is triggered. An output of -1 specifies that the neuron did not get triggered.
“sgn” stands for sign function with output +1 or -1.

Error in Perceptron

In the Perceptron Learning Rule, the predicted output is compared with the known output. If it does not match, the error is propagated backward to allow weight adjustment to happen.
Let us discuss the decision function of Perceptron in the next section.

Perceptron: Decision Function

A decision function φ(z) of Perceptron is defined to take a linear combination of x and w vectors.

$$\boldsymbol{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}$$

The value z in the decision function is given by:

$$\mathrm{z} = w_1 x_1 + \ldots + w_m x_m$$

The decision function is +1 if z is greater than a threshold θ, and it is -1 otherwise.

$$\phi(\mathrm{z}) = \begin{cases} 1 & if\ z \geq \theta \\ -1 & otherwise \end{cases}$$

This is the Perceptron algorithm.

Bias Unit

For simplicity, the threshold θ can be brought to the left and represented as w0x0, where w0= -θ and x0= 1.

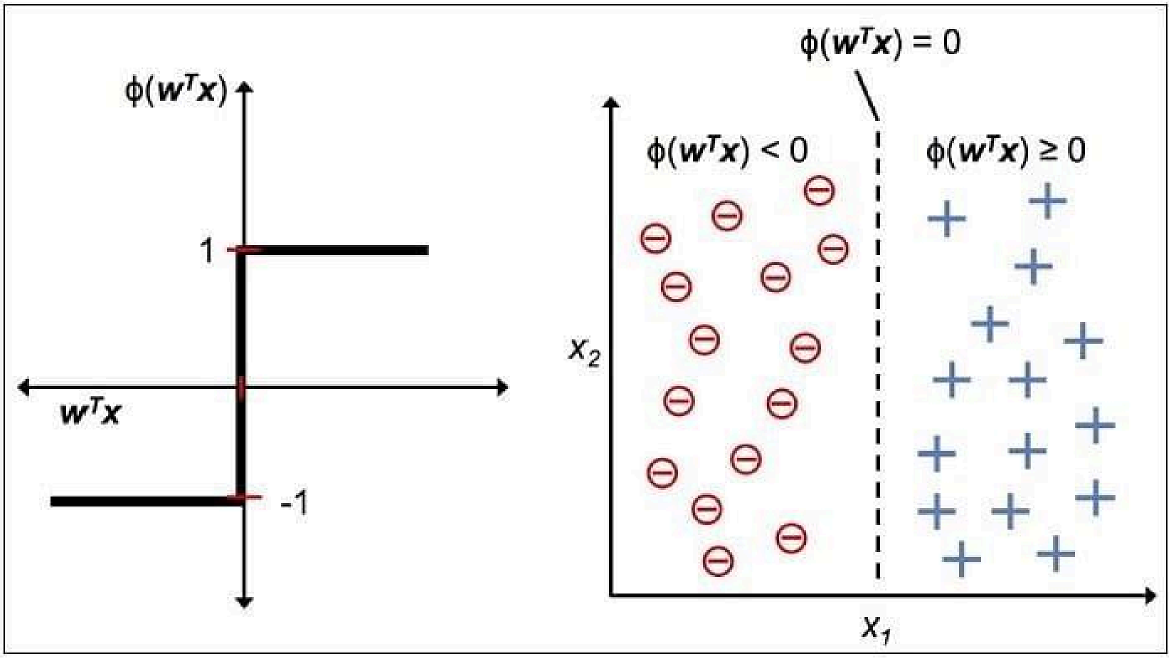
$$\mathrm{z} = w_0 x_0 + w_1 x_1 + \ldots + w_m x_m = \boldsymbol{w}^T \boldsymbol{x}$$

The value w0 is called the bias unit.
The decision function then becomes:

$$\phi(z) = \begin{cases} 1 & if\ z \geq 0 \\ -1 & otherwise \end{cases}$$

Output:

The figure shows how the decision function squashes wTx to either +1 or -1 and how it can be used to discriminate between two linearly separable classes.



Perceptron at a Glance

Perceptron has the following characteristics:

- Perceptron is an algorithm for Supervised Learning of single layer binary linear classifiers.
- Optimal weight coefficients are automatically learned.
- Weights are multiplied with the input features and decision is made if the neuron is fired or not.
- Activation function applies a step rule to check if the output of the weighting function is greater than zero.
- Linear decision boundary is drawn enabling the distinction between the two linearly separable classes +1 and -1.
- If the sum of the input signals exceeds a certain threshold, it outputs a signal; otherwise, there is no output.

Types of activation functions include the sign, step, and sigmoid functions.

Implement Logic Gates with Perceptron

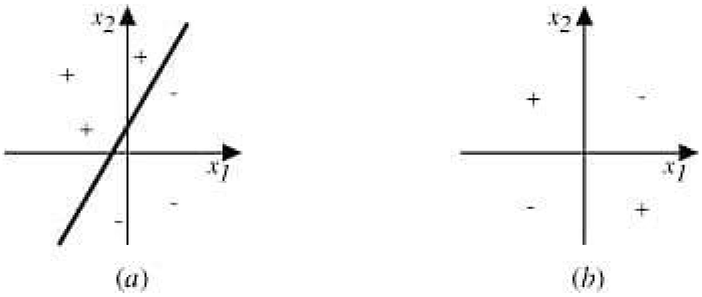
Perceptron - Classifier Hyperplane

The Perceptron learning rule converges if the two classes can be separated by the linear hyperplane. However, if the classes cannot be separated perfectly by a linear classifier, it could give rise to errors.

As discussed in the previous topic, the classifier boundary for a binary output in a Perceptron is represented by the equation given below:

$$\vec{w} \cdot \vec{x} = 0,$$

The diagram above shows the decision surface represented by a two-input Perceptron.



Observation:

- In Fig(a) above, examples can be clearly separated into positive and negative values; hence, they are linearly separable. This can include logic gates like AND, OR, NOR, NAND.
- Fig (b) shows examples that are not linearly separable (as in an XOR gate).
- Diagram (a) is a set of training examples and the decision surface of a Perceptron that classifies them correctly.
- Diagram (b) is a set of training examples that are not linearly separable, that is, they cannot be correctly classified by any straight line.
- X1 and X2 are the Perceptron inputs.

In the next section, let us talk about logic gates.

What is Logic Gate?

Logic gates are the building blocks of a digital system, especially neural networks. In short, they are the electronic circuits that help in addition, choice, negation, and combination to form complex circuits. Using the logic gates, Neural Networks can learn on their own without you having to manually code the logic. Most logic gates have two inputs and one output.

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

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

- AND

- NAND
- OR
- NOR
- NOT
- XOR
- XNOR

Implementing Basic Logic Gates With Perceptron

The logic gates that can be implemented with Perceptron are discussed below.

1. AND

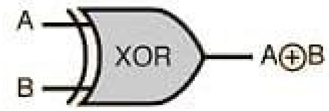
If the two inputs are TRUE (+1), the output of Perceptron is positive, which amounts to TRUE.
This is the desired behavior of an AND gate.
x1= 1 (TRUE), x2= 1 (TRUE)
w0 = -.8, w1 = 0.5, w2 = 0.5
=> o(x1, x2) => -.8 + 0.5*1 + 0.5*1 = 0.2 > 0

2. OR

If either of the two inputs are TRUE (+1), the output of Perceptron is positive, which amounts to TRUE.
This is the desired behavior of an OR gate.
x1 = 1 (TRUE), x2 = 0 (FALSE)
w0 = -.3, w1 = 0.5, w2 = 0.5
=> o(x1, x2) => -.3 + 0.5*1 + 0.5*0 = 0.2 > 0

3. XOR

A XOR gate, also called as Exclusive OR gate, has two inputs and one output.



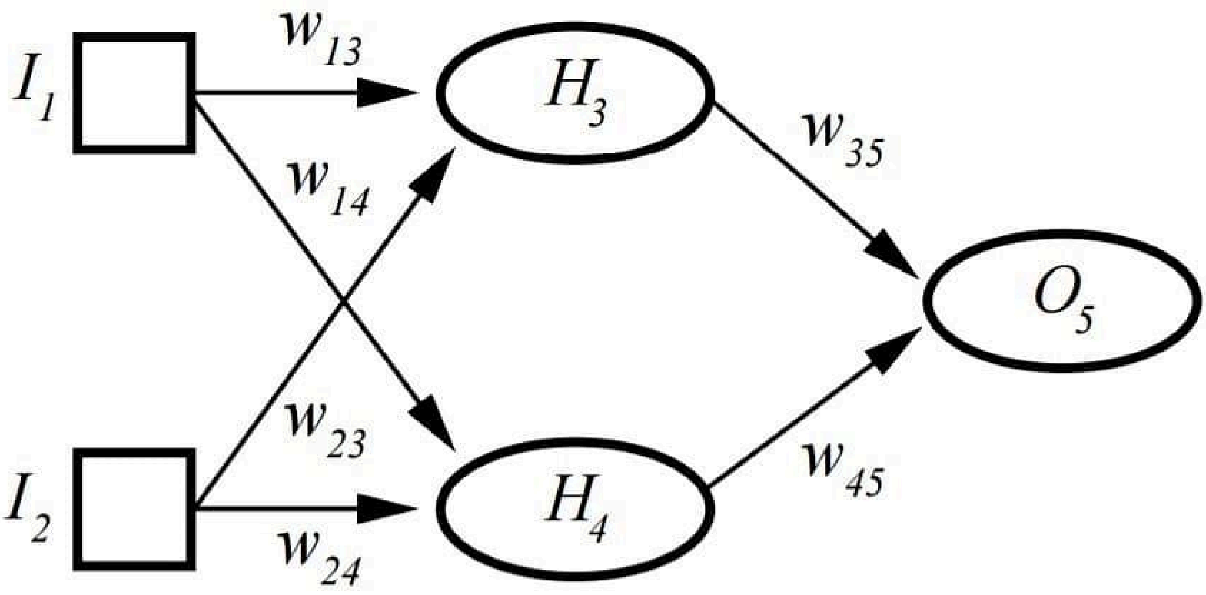
The gate returns a TRUE as the output if and ONLY if one of the input states is true.
XOR Truth Table

Input		Output
A	B	Y
0	0	0

0	1	1
1	0	1
1	1	0

XOR Gate with Neural Networks

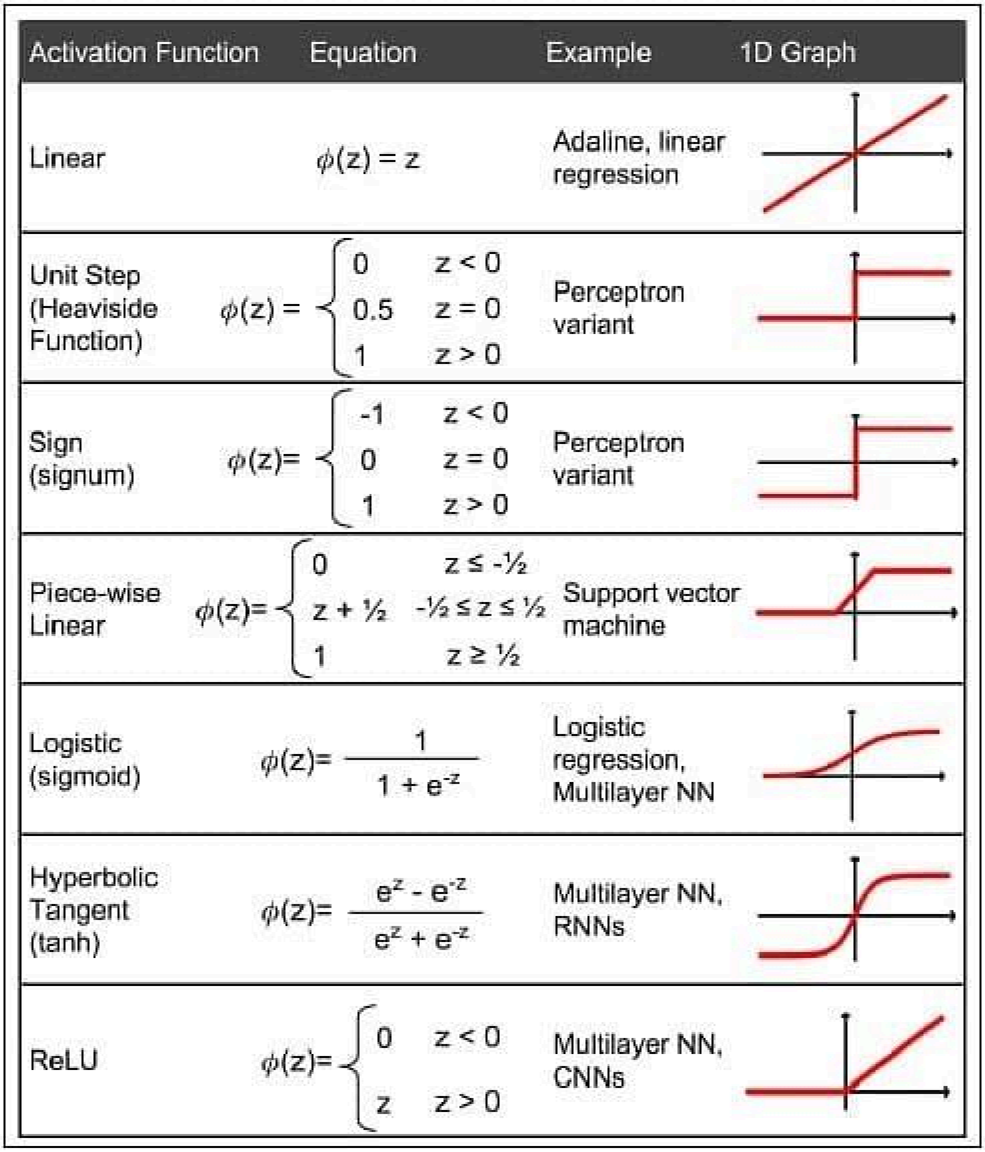
Unlike the AND and OR gate, an XOR gate requires an intermediate hidden layer for preliminary transformation in order to achieve the logic of an XOR gate.



An XOR gate assigns weights so that XOR conditions are met. It cannot be implemented with a single layer Perceptron and requires Multi-layer Perceptron or MLP.
H represents the hidden layer, which allows XOR implementation.
I1, I2, H3, H4, O5are 0 (FALSE) or 1 (TRUE)
t3= threshold for H3; t4= threshold for H4; t5= threshold for O5
H3= sigmoid (I1*w13+ I2*w23–t3); H4= sigmoid (I1*w14+ I2*w24–t4)
O5= sigmoid (H3*w35+ H4*w45–t5);
Next up, let us learn more about the Sigmoid activation function!

Activation Functions at a Glance

Various activation functions that can be used with Perceptron are shown below:



Activation Function	Equation	Example	1D Graph
Linear	$\phi(z) = z$	Adaline, linear regression	
Unit Step (Heaviside Function)	$\phi(z) = \begin{cases} 0 & z < 0 \\ 0.5 & z = 0 \\ 1 & z > 0 \end{cases}$	Perceptron variant	
Sign (signum)	$\phi(z)= \begin{cases} -1 & z < 0 \\ 0 & z = 0 \\ 1 & z > 0 \end{cases}$	Perceptron variant	
Piece-wise Linear	$\phi(z)= \begin{cases} 0 & z \leq -\frac{1}{2} \\ z + \frac{1}{2} & -\frac{1}{2} \leq z \leq \frac{1}{2} \\ 1 & z \geq \frac{1}{2} \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z)= \dfrac{1}{1 + e^{-z}}$	Logistic regression, Multilayer NN	
Hyperbolic Tangent (tanh)	$\phi(z)= \dfrac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multilayer NN, RNNs	
ReLU	$\phi(z)= \begin{cases} 0 & z < 0 \\ z & z > 0 \end{cases}$	Multilayer NN, CNNs	

The activation function to be used is a subjective decision taken by the data scientist, based on the problem statement and the form of the desired results. If the learning process is slow or has vanishing or exploding gradients, the data scientist may try to change the activation function to see if these problems can be resolved.

Future of Perceptron

With the increasing popularity and usage of Machine Learning, the future of Perceptron seems significant and prospectus. It helps to interpret data by building innate patterns and applying them shortly. Coding is continuously evolving in this era, and the end of perceptron technology will continue to support and facilitate analytical behavior in machines that will add further efficiency to modern computers.

Summary

Let us summarize what we have learned in this tutorial:

- An artificial neuron is a mathematical function conceived as a model of biological neurons, that is, a neural network.
- A Perceptron is a neural network unit that does certain computations to detect features or business intelligence in the input data. It is a function that maps its input “x,” which is multiplied by the learned weight coefficient, and generates an output value ”f(x).
- ”Perceptron Learning Rule states that the algorithm would automatically learn the optimal weight coefficients.
- Single layer Perceptrons can learn only linearly separable patterns.
- Multilayer Perceptron or feedforward neural network with two or more layers have the greater processing power and can process non-linear patterns as well.
- Perceptrons can implement Logic Gates like AND, OR, or XOR.

Summary

The perceptron is a simple and interpretable linear classifier, and the perceptron training rule is an effective method for adjusting weights to correctly classify linearly separable data. Although it has limitations, the perceptron laid the groundwork for modern neural networks by introducing the concept of weight adjustments based on error, which is foundational in more advanced learning algorithms like backpropagation.

Training in-thresholded perceptron using Delta rule:

The Delta rule (also known as the Least Mean Squares (LMS) rule) is used to train an unthresholded perceptron. Unlike the basic perceptron training rule, which relies on a thresholded output, the Delta rule is applied to a perceptron with a continuous output that can take any value (usually between 0 and 1, if we use a sigmoid activation function). This rule minimizes the difference between the actual and desired outputs, and it forms the foundation for training multilayer perceptrons.

The Gradient Descent and The Delta Rule for training a perceptron and its implementation using python.

Why Gradient Descent ?

As we have discussed earlier, the perceptron training rule works for the training samples of data that are linearly separable. Another limitation is that if there are multiple local minima, then the previous rule might converge to a local minima instead of global minima. To overcome these limitations, we are going to use gradient descent for training our perceptron.

> গ্রেডিয়েন্ট ডিসেন্ট হল একটি গাণিতিক অ্যালগরিদম যা ফাংশনের নেতিবাচক গ্রেডিয়েন্টের দিকে পুনরাবৃত্তভাবে চলে যাওয়ার মাধ্যমে একটি ফাংশনের সর্বনিম্ন মান খুঁজে পায়|

How does it work ?

The idea behind the gradient descent or the delta rule is that we search the hypothesis space of all possible weight vectors to find the best fit for our training samples. Gradient descent acts like a base for Back Propagation algorithms, which we will discuss in upcoming posts.
Delta Rule can be understood by looking at it as training and threshold perceptron which is trained using gradient descent . The linear combination of weights and the inputs associated with them acts as an input to activation function same as in the previous one.
O(x)= w.x
Before going into the activation function, we need to know how do we calculate the training error in the weights in case of misclassification. In Gradient Descent, we commonly use half the squared difference between the ouput and obtained value as the training error for our hypotheses.

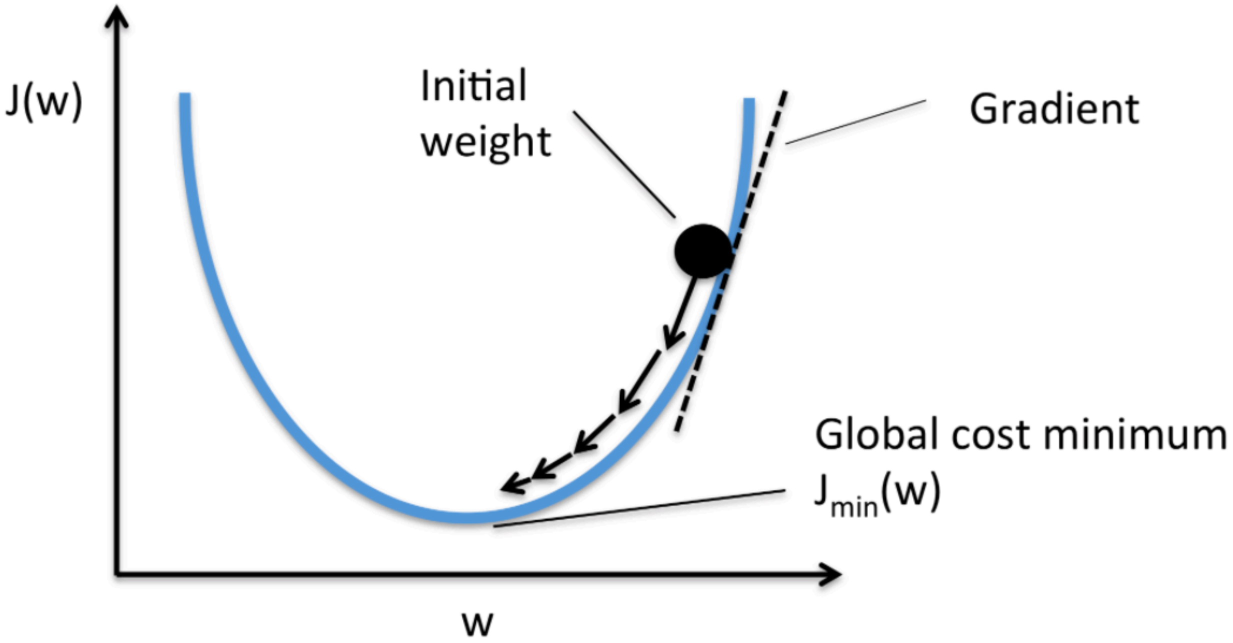
$$E(\vec{w}) = \frac{1}{2} \sum_{d \epsilon D} (t_d - o_d)^2$$

Training error for Gradient Descent

- D is a set of training samples.
- t is the target output for training example ‘d’.
- o is the output of a linear unit for training example ‘d’.

We would use the same unit step function as the activation function for this example too.

Visualizing the Weight vectors and their Error values:



Gradient Descent

- w represents all possible weight vectors.
- J(w) represents the error values with respect to weight vectors.
- Parabolic path of error surface, which has a global minima at which the weight vectors are most suited for the training samples.

Derivation of Gradient Descent Rule

In Gradient Descent, we calculate the steepest descent along the error surface for finding local minima. For this we need to calculate the derivative of E with respect to the weight vector. This vector derivative is called Gradient of E with respect to weight vector w. The training rule can now be represented as :

$$\vec{w} = \vec{w} + \Delta\vec{w}$$

Fig 1.0

$$\Delta\vec{w} = -\eta \, \nabla E(\vec{w})$$

Fig 1.1

where E(w) is given by differentiating the training errors of data samples with their corresponding weight vectors. Substituting the whole thing in the above equation gives us:

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i}\frac{1}{2}\sum_{d\epsilon D}(t_d - o_d)^2$$

Fig 1.2

Differentiating the above equation (Fig 1.2)and substituting o = w.x in it would give us the result as follows:

$$\frac{\partial E}{\partial w_i} = \sum_{d\epsilon D}(t_d - o_d)\ (-x_{id})$$

Therefore, by substituting Fig 1.3 in Fig 1.1 we get the final weight updation rule of Gradient Descent:

$$\Delta w_i = \eta \sum_{d \in D}(t_d - o_d)\, x_{id}$$

WEIGHT UPDATION RULE IN GRADIENT DESCENT

We have arrived at our final equation on how to update our weights using the delta rule. One more key difference is that, in perceptron rule we modify the weights after training all the samples, but in delta rule we update after every misclassification, making the chance of reaching global minima high.

Algorithm:

- Initialize the weight vector.
- For each training sample in the dataset, apply the activation function and if any error occurs, update the weight according to the rule.
- Repeat it for a finite number of epochs, to make it more accurate.

Why Do We Need Non-Linearity?

Non-linearity is essential in neural networks to capture complex patterns in data that cannot be separated by a simple linear decision boundary. Here are the primary reasons why non-linearity is necessary:

1. Non-Linearly Separable Data:
    - Many real-world problems, such as image recognition, language processing, and complex classification tasks, involve data that is not linearly separable. A purely linear model would not be able to solve these problems effectively, as it would be restricted to creating linear decision boundaries.
2. Ability to Learn Complex Patterns:
    - Non-linear activation functions, such as the sigmoid, ReLU, and tanh, introduce non-linearity into the network, enabling it to model intricate relationships in data.

These functions allow the network to learn complex mappings between inputs and outputs, creating curved or even multi-dimensional decision boundaries as needed.

3. Stacking Layers Effectively:
    - In a multi-layer neural network, applying non-linear activation functions between layers allows each layer to transform the input data in different ways. Without non-linearity, multiple layers would simply collapse into a single linear transformation, rendering deep networks ineffective. Non-linear functions allow each layer to build on the previous one, creating progressively more abstract and high-level representations of the data.

Types of Network Structures: Feedforward Networks and Recurrent Networks

1. Feedforward Networks
    - Structure: In a feedforward neural network, information flows in a single direction—from input to output—without any feedback loops or recurrent connections.
    - Layers: These networks typically consist of an input layer, one or more hidden layers, and an output layer.
    - Activation Functions: Each neuron applies an activation function to its input before passing it to the next layer.
    - Usage: Feedforward networks are widely used for tasks such as image classification, object recognition, and other applications where input-output relationships are static and do not depend on any sequence or temporal dependencies.
2. Advantages:
    - Simple and easy to implement and train.
    - Effective for tasks that require a straightforward mapping from input to output.
3. Limitations:
    - Not suitable for sequential or temporal data, where context from previous inputs might influence current decisions (e.g., in language processing).
4. Recurrent Neural Networks (RNNs)
    - Structure: Recurrent neural networks have connections that allow information to persist across steps in a sequence. They introduce feedback loops, where the output of a neuron can influence its future input, enabling the network to maintain information over time.
    - Hidden State: RNNs maintain a hidden state that gets updated at each time step, allowing them to remember information from previous inputs, which is crucial for sequential data.

- Usage: RNNs are commonly used for tasks where temporal dependencies are critical, such as speech recognition, language modeling, time series prediction, and any data involving sequences.

5. Advantages:
    - Suitable for handling sequential and time-dependent data.
    - Can remember information over multiple time steps, providing context to the model.
6. Limitations:
    - Training RNNs can be challenging due to issues like the vanishing gradient problem, especially when trying to remember information over long sequences.
    - RNNs can be computationally expensive, as they require iterative processing of each time step.
7. Variants like Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs) were developed to address these issues, allowing RNNs to handle longer dependencies more effectively.

নিউরাল নেটওয়ার্কগুলিতে অ-রৈখিকতা গুরুত্বপূর্ণ কারণ এটি তাদের জটিল নিদর্শনগুলি শিখতে এবং মডেল করতে দেয় , যেমন প্রাকৃতিক ভাষা প্রক্রিয়াকরণ (NLP), সময়-সিরিজ বিশ্লেষণ এবং অনুক্রমিক ডেটা ভবিষ্যদ্বাণীতে পাওয়া যায়।

Summary

Non-linearity in neural networks enables them to model complex and non-linear relationships in data, which is essential for solving most real-world problems. Feedforward networks are effective for static input-output mappings, while recurrent networks excel at capturing temporal dependencies in sequential data. Together, these network structures provide the foundation for designing neural networks tailored to specific tasks and data types.

ফিডফরওয়ার্ড নিউরাল নেটওয়ার্কগুলি এমন কাজের জন্য উপযুক্ত যা একটি একক আউটপুটের পূর্বাভাস জড়িত, যেমন চিত্র শ্রেণীবিভাগ বা প্রাকৃতিক ভাষা প্রক্রিয়াকরণ। পৌনঃপুনিক নিউরাল নেটওয়ার্কগুলি এমন কাজের জন্য উপযুক্ত যা অনুক্রমিক ডেটা জড়িত, যেমন স্পিচ রিকগনিশন বা মেশিন অনুবাদ।

Generalization, overfitting, and stopping criterion, overcoming the overfitting problem using a set of validation data.

In machine learning and neural networks, achieving a model that can generalize well to new data is crucial. This leads to the concepts of generalization, overfitting, and stopping criteria during training. Let's explore each of these concepts and the strategies used to address overfitting.

1. Generalization

Generalization is the ability of a neural network to perform well on new, unseen data (called test data) after being trained on a specific dataset (called training data). A model that generalizes well captures the underlying patterns in the data rather than memorizing the training examples.

2. Overfitting

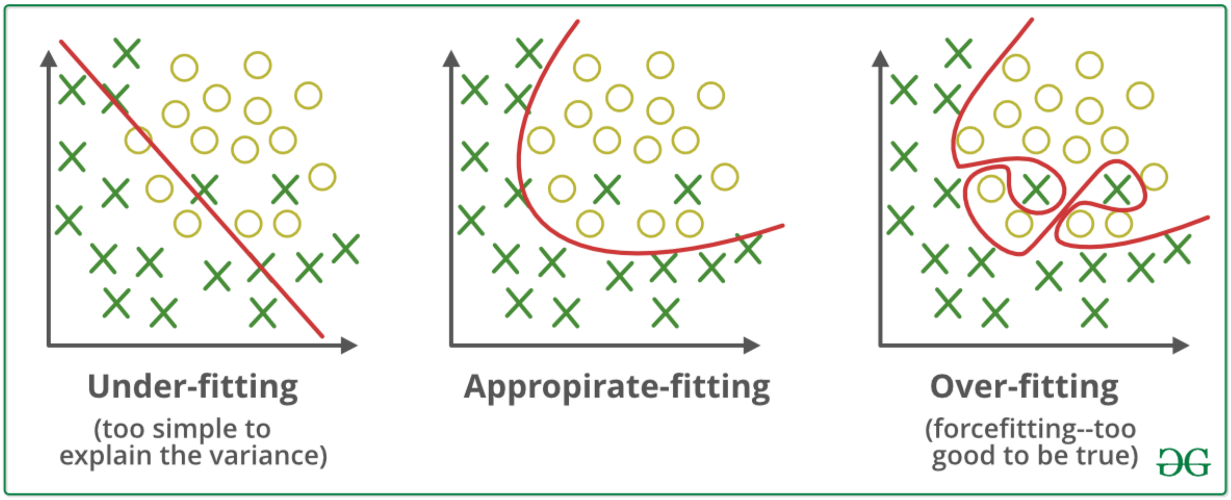
Overfitting occurs when a model learns the training data too well, capturing noise and irrelevant details along with the underlying pattern. This results in high accuracy on the training data but poor performance on new data.

- Causes of Overfitting:
    - Complex Model: When the model has too many parameters (e.g., too many layers or neurons), it can "memorize" the training data.
    - Insufficient Training Data: With a small amount of training data, the model may learn specific examples rather than general patterns.
    - High Training Epochs: Training the model for too many epochs can result in overfitting, as the model starts to adapt to the peculiarities of the training data.

How to avoid the Overfitting in Model

Both overfitting and underfitting cause the degraded performance of the machine learning model. But the main cause is overfitting, so there are some ways by which we can reduce the occurrence of overfitting in our model.

- Cross-Validation
- Training with more data
- Removing features
- Early stopping the training
- Regularization
- Ensembling


Under-fitting
(too simple to explain the variance)
Appropirate-fitting
Over-fitting
(forcefitting--too good to be true)

3. Stopping Criterion

The stopping criterion determines when to stop training a neural network. Without a proper stopping criterion, a model may continue to train until it overfits the training data. Common stopping criteria include:

- Early Stopping: Monitoring the model's performance on a separate set of data (the validation set) during training. When performance on the validation set stops improving, training is halted to prevent overfitting.
- Fixed Number of Epochs: Setting a predetermined number of epochs based on prior knowledge or experiments. However, this method is less adaptive and may not prevent overfitting effectively.
- Validation-Based Stopping: Some algorithms automatically track the validation error, and training stops when this error stops decreasing or starts increasing.

4. Overcoming the Overfitting Problem

To address overfitting, several techniques can be applied. Using a validation set is key to most of these methods, as it provides feedback on how well the model generalizes beyond the training data.

Methods to Prevent Overfitting:

1. Validation Set:
   - Split the data into training, validation, and test sets. During training, the model is evaluated on the validation set after each epoch, enabling us to detect and prevent overfitting.
   - Early stopping is applied when the performance on the validation set plateaus or worsens, indicating the model has reached its best state for generalization.
2. Regularization Techniques:

   - L1 and L2 Regularization: Adding a penalty to the loss function for large weights, which discourages complex models that can overfit the training data.
   - Dropout: Randomly dropping a subset of neurons (setting their activations to zero) during each training step, which forces the network to learn redundant, robust features that generalize better.

3. Data Augmentation:
   - For image data, data augmentation can involve rotating, flipping, or adding noise to the images. This technique effectively increases the amount and variability of training data, helping the model generalize better.
4. Reduce Model Complexity:
   - Simplify the model architecture by reducing the number of layers or neurons. Smaller models are less prone to overfitting, especially when data is limited.
5. Cross-Validation:
   - Cross-validation (e.g., k-fold cross-validation) involves dividing the data into multiple subsets and training/testing the model multiple times with different subsets. This provides a more robust measure of the model’s generalization performance.

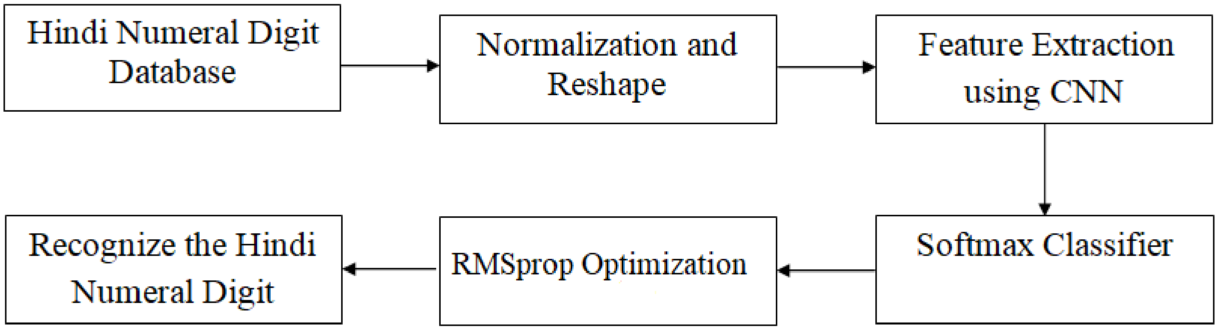
Summary

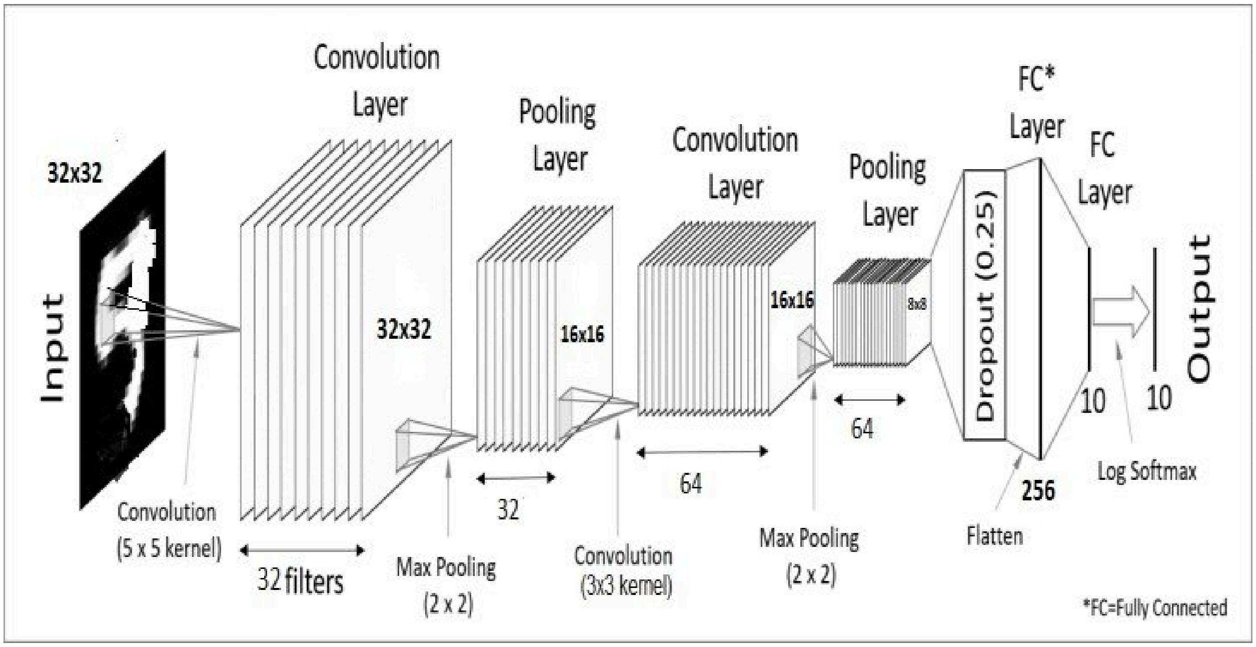
- Generalization is the model’s ability to perform well on unseen data.
- Overfitting is when the model memorizes training data rather than learning patterns, causing poor generalization.
- Stopping Criterion helps prevent overfitting, with early stopping being one of the most common techniques.
- Validation Data and regularization methods like dropout, L1/L2 penalties, and data augmentation are essential in preventing overfitting, helping the model generalize effectively.

These methods work together to improve the model's robustness and ensure it captures relevant patterns that generalize well across new data.

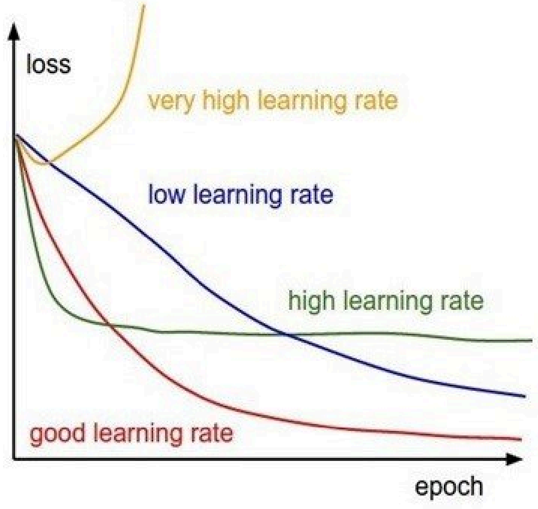
ANN architecture for handwritten digit recognition

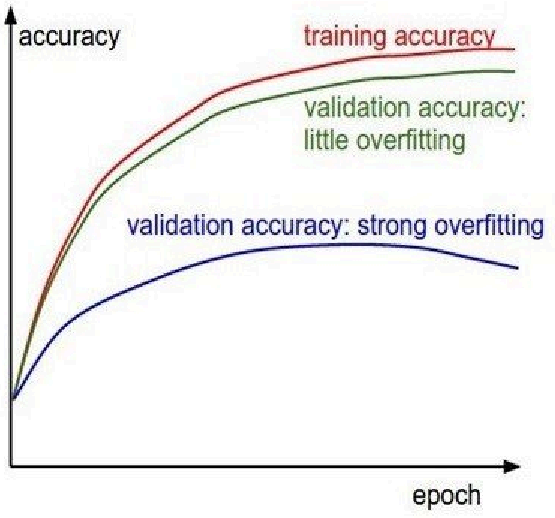
For handwritten digit recognition using an Artificial Neural Network (ANN), a common approach is to design a network with an input layer, one or more hidden layers, and an output layer. Here’s a simplified example:


Hindi Numeral Digit Database
Normalization and Reshape
Feature Extraction using CNN
Softmax Classifier
RMSprop Optimization
Recognize the Hindi Numeral Digit



The overall structural design of the CNN Model of our proposed system with different layers. Convolutional Neural Network (CNNs) with RMSprop for digits recognition is trained on Hindi Handwritten numerals Dataset. Where keras API is used with Tensorflow as a backend. A total of 7 layers are used on which the first and third are convolutional(Conv2D) layers, second and fourth are important layer in CNN is the pooling (MaxPool2D) layers, Flatten layer and the last two fully connected Dense layers which is just artificial and neural networks (ANN) classifier. The first layer is the convolutional we used is Conv2D with learnable filters of size 32 filters for first and 64 filters for the last ones in the model. Each filter transforms a part of the image by defining the kernel size using the kernel filter. The kernel filter matrix of size 5x5 is applied on the whole image. Filters can be seen as a transformation of the image with feature maps. Padding concept here is "same" that means it results in padding the input such that the output has the same length as the original input [8][11]. The figure 4 is shown below how the padding happens.





1. Input Representation:

- Input Layer:
    - The input layer represents the pixel values of a 28x28 grayscale image of a digit (as in the MNIST dataset).
    - Each pixel has a grayscale value between 0 and 255, which can be normalized to a range between 0 and 1.
    - With 28x28 pixels, the input layer consists of 784 neurons (one for each pixel).

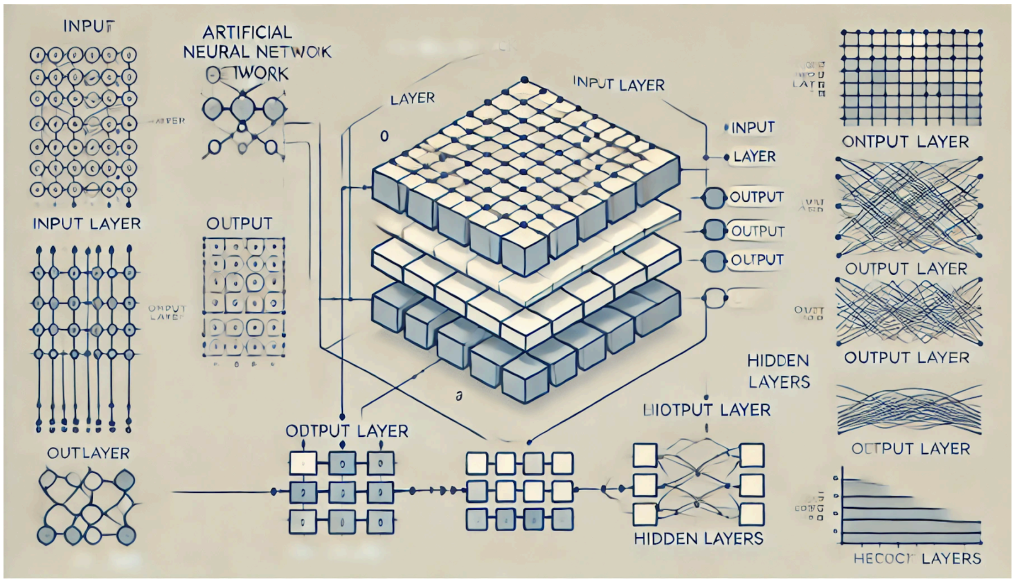
2. Output Representation:

- Output Layer:
    - The output layer represents the digits from 0 to 9.
    - This layer has 10 neurons, each corresponding to one digit.

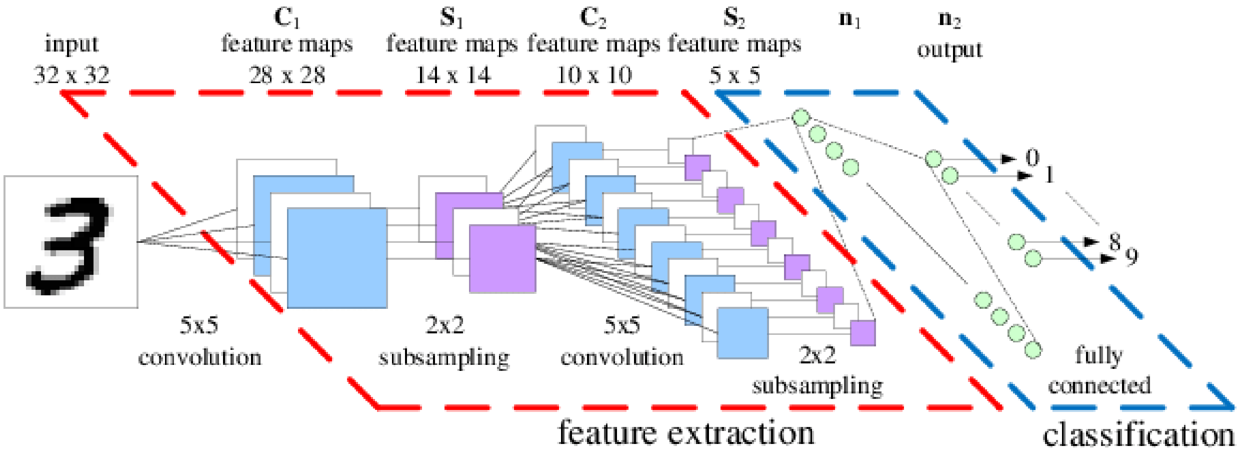
- ○ A softmax activation function is commonly used in the output layer to provide probabilities for each class (0-9), where the neuron with the highest probability indicates the predicted digit.

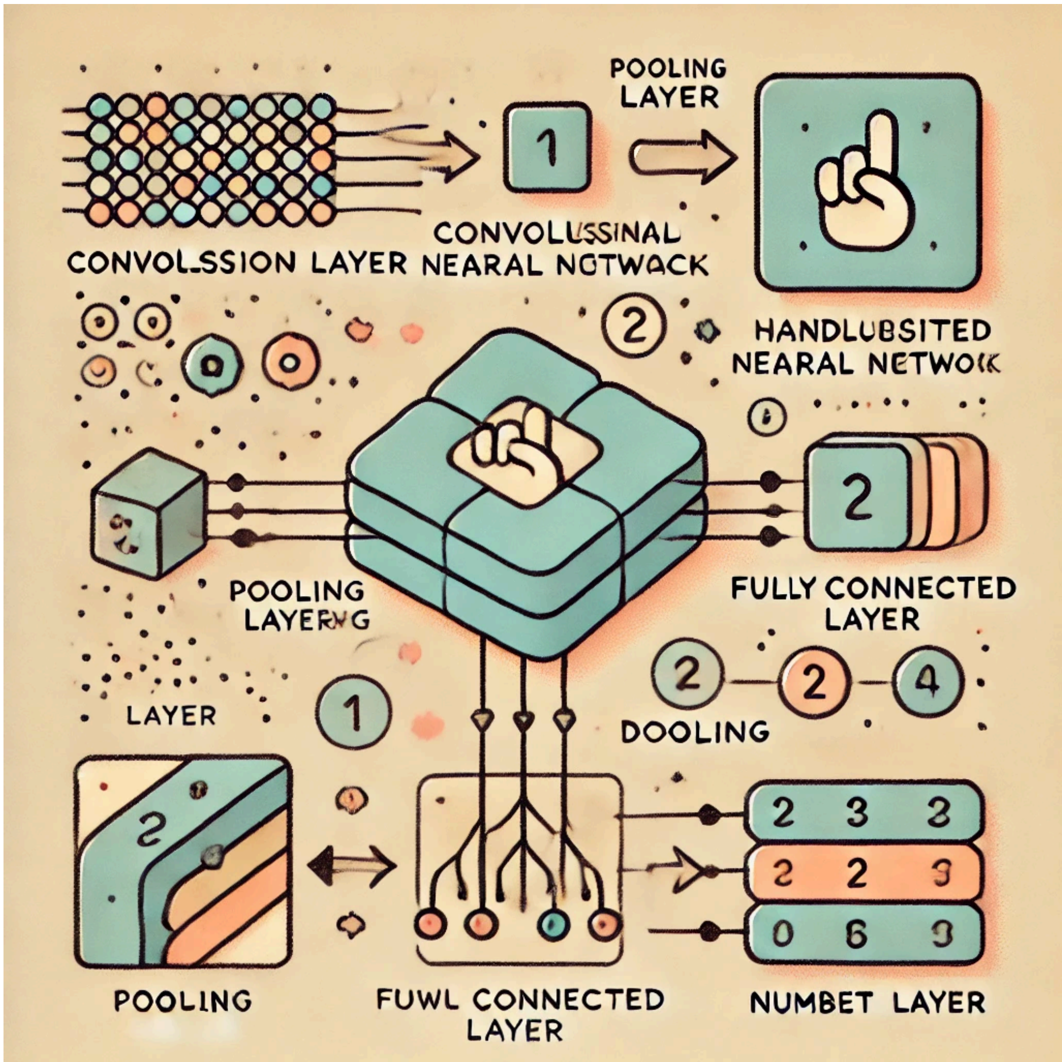
3. Block Diagram of the Network:

Here’s a general block diagram of an ANN architecture for handwritten digit recognition:



A block diagram illustrating CNN applied to handwritten digit recognition task





Need for automatic feature learning, difference between the conventional feed-forward neural networks and CNN, role of convolution layer in CNN, An example of 2D convolution, function of pooling layer

Here’s a brief overview of each of those concepts to help convey their importance and functionality:

1. Need for Automatic Feature Learning

- Manual Feature Extraction: Conventional methods for image analysis required manual feature extraction, which is labor-intensive and may miss subtle patterns.
- Automatic Feature Learning: CNNs automatically learn important features (e.g., edges, textures, and shapes) directly from raw data. This enhances performance in complex tasks like image classification without needing predefined features.

2. Difference Between Feed-Forward Neural Networks and CNNs

- Feed-Forward Neural Networks (FNNs): In FNNs, each neuron is fully connected to the next layer. These networks work well for structured data but struggle with high-dimensional data like images due to extensive parameters and computational cost.
- Convolutional Neural Networks (CNNs): CNNs introduce convolutional and pooling layers, reducing the number of parameters by sharing weights across spatial locations. This makes CNNs more efficient and effective for image and spatial data by recognizing patterns and local features.

3. Role of the Convolutional Layer in CNNs

- The Convolutional Layer acts as a feature extractor. Filters (small matrices) slide over the input image, identifying local patterns such as edges, corners, or textures.
- Each filter produces a feature map, emphasizing the detected feature, and multiple filters enable the CNN to learn diverse features automatically.

4. Example of 2D Convolution

- In a 2D Convolution:
    1. A filter (e.g., 3x3 matrix) slides across the image matrix.
    2. Each filter position produces a new value by multiplying corresponding elements in the filter and input, then summing the results.
    3. This produces a feature map that highlights specific patterns in the image.
- This process allows CNNs to retain spatial relationships and recognize shapes or textures at different positions in the image.

5. Function of the Pooling Layer

- Pooling Layers downsample feature maps to reduce their spatial dimensions, lowering computational load and improving robustness against variations.
- Commonly used methods include Max Pooling (selects the maximum value from each region) and Average Pooling (takes the average of each region). This helps CNNs focus on the most prominent features while discarding noise and preserving key information.

These principles combined allow CNNs to efficiently process complex, high-dimensional data like images, making them foundational in fields such as image recognition, object detection, and more.

Multiple Choice Questions (MCQs)

1. What is the primary biological motivation for Artificial Neural Networks (ANNs)?
    - A) High computing power
    - B) Neuron structure in the human brain
    - C) Memory storage
    - D) Machine learning applications
    - Answer: B
2. Who introduced the first mathematical model of a neuron?
    - A) Rosenblatt
    - B) McCulloch and Pitts
    - C) Hebb
    - D) Rumelhart
    - Answer: B
3. Which of the following functions can serve as an activation function?
    - A) Logarithmic function
    - B) Step function
    - C) Exponential function
    - D) Subtraction function
    - Answer: B
4. The Sigmoid function outputs values between:
    - A) -1 and 1
    - B) 0 and 1
    - C) -1 and 0
    - D) 0 and infinity
    - Answer: B
5. Which function in a perceptron decides whether a neuron will "fire"?
    - A) Loss function
    - B) Activation function
    - C) Cost function
    - D) Derivative function

   - Answer: B

6. What is the Perceptron primarily used for?
   - A) Nonlinear classification
   - B) Linear classification
   - C) Time-series analysis
   - D) Dimensionality reduction
   - Answer: B
7. The OR function of two inputs in a perceptron requires which type of activation threshold?
   - A) 0
   - B) Greater than 1
   - C) Less than 0
   - D) 1
   - Answer: D
8. The Delta rule is used for training:
   - A) Thresholded perceptrons
   - B) Unthresholded perceptrons
   - C) Recurrent networks
   - D) Convolutional layers
   - Answer: B
9. Why do we need non-linearity in neural networks?
   - A) To make the network simpler
   - B) To allow the network to model complex patterns
   - C) To increase computation speed
   - D) For linear separation only
   - Answer: B
10. XOR function in perceptron learning can be represented by:
    - A) A single-layer perceptron
    - B) A multi-layer perceptron
    - C) A threshold perceptron
    - D) A linear perceptron
    - Answer: B
11. In which type of network do connections form loops?
    - A) Feed-forward networks
    - B) Convolutional neural networks
    - C) Recurrent networks
    - D) Multi-layer networks
    - Answer: C
12. Backpropagation is mainly used in:
    - A) Training recurrent networks

- B) Optimization of weights in multi-layer networks
- C) Feature extraction in CNNs
- D) Data normalization
- Answer: B

13. To prevent overfitting, we use:
    - A) More hidden layers
    - B) Validation data
    - C) Increased training data only
    - D) Higher learning rates
    - Answer: B
14. CNNs are best suited for tasks involving:
    - A) Time series data
    - B) Image and spatial data
    - C) Text analysis
    - D) Numerical datasets
    - Answer: B
15. Pooling layers in CNNs primarily serve to:
    - A) Increase spatial dimensions
    - B) Reduce dimensionality
    - C) Increase computation time
    - D) Add filters
    - Answer: B
16. A function that can represent AND in a perceptron has:
    - A) A positive threshold
    - B) A high negative weight
    - C) An activation threshold of 1
    - D) A non-zero threshold
    - Answer: C
17. Overfitting in a neural network occurs when:
    - A) The model learns noise as patterns
    - B) The model uses high learning rates
    - C) Training data is insufficient
    - D) Only output neurons are over-activated
    - Answer: A
18. Which technique is often used to overcome overfitting?
    - A) Using more complex models
    - B) Increasing learning rate
    - C) Early stopping
    - D) Using unlabelled data
    - Answer: C

19. A simple neural network structure for handwritten digit recognition usually starts with:
    - A) Random noise
    - B) Convolutional layers
    - C) Fully connected layers
    - D) Recurrent connections
    - Answer: B
20. The linear separator equation in perceptron learning is:
    - A) w.x = b
    - B) y = mx + c
    - C) w.x + b = 0
    - D) f(x) = x
    - Answer: C
21. In McCulloch and Pitts' model, neuron firing is represented by:
    - A) Input layer activation
    - B) Binary threshold
    - C) Continuous threshold
    - D) Noise reduction
    - Answer: B
22. Which activation function outputs 0 or 1?
    - A) ReLU
    - B) Sigmoid
    - C) Threshold function
    - D) Tanh
    - Answer: C

Short Answer Questions (SAQs)

1. Describe the biological inspiration behind ANNs.
2. Explain McCulloch and Pitts' mathematical model of a neuron.
3. Define an activation function and its importance.
4. What is a perceptron, and how is it used as a linear classifier?
5. Briefly describe the Delta rule for perceptron training.
6. Why is non-linearity essential in neural networks?
7. Differentiate between feed-forward and recurrent networks.
8. What is the XOR problem in perceptron learning?
9. Explain the need for hidden layers in neural networks.
10. What is the role of pooling in CNNs?
11. How does the convolutional layer function in CNNs?
12. Define overfitting and its implications in neural networks.
13. Describe how CNNs differ from conventional feed-forward networks.

14. Explain the concept of early stopping in training neural networks.
15. How does backpropagation work in neural networks?
16. Define the sigmoid activation function and its range.
17. Describe the basic structure of a CNN for handwritten digit recognition.
18. What is a linear separator, and how does it apply in perceptron learning?
19. Explain the concept of automatic feature learning in CNNs.
20. What is the main challenge when applying a single-layer perceptron to solve non-linear problems?

5-Mark Questions

1. Explain the biological motivation for artificial neural networks and discuss the significance of automatic feature learning.
2. Describe the McCulloch and Pitts model of a neuron, including the concept of activation functions with examples.
3. Define perceptron training and illustrate how the perceptron training rule is applied.
4. Derive the Delta rule for training an unthresholded perceptron, and explain its function in weight adjustment.
5. Explain why non-linearity is needed in neural networks, and differentiate between feed-forward and recurrent network structures.
6. Describe the backpropagation algorithm and how it is used to train multi-layer feed-forward neural networks.
7. Explain overfitting in neural networks, and describe strategies for overcoming it, such as using a validation dataset.
8. Illustrate with examples the representational power of perceptrons in implementing logical functions like AND and OR.
9. Draw and explain a CNN structure for handwritten digit recognition, outlining input and output representations.
10. Explain the difference between CNNs and conventional feed-forward neural networks, emphasizing the role of convolution and pooling layers in CNNs.

Ethical Issues in Artificial Intelligence (AI)

Introduction

Artificial Intelligence (AI) means machines or computer systems that can think, learn, and make decisions like humans. AI is used in many areas such as healthcare, education, banking, transport, and mobile apps. Although AI is very useful, it also creates some ethical problems. Ethical issues mean what is right or wrong when using AI.

1. Bias and Unfair Decisions

AI systems learn from data. If the data is unfair or incorrect, AI can make biased decisions.
For example, an AI system may reject job applications or loans unfairly for some people. This is wrong and unethical. AI should treat everyone equally and fairly.

2. Privacy and Data Safety

AI needs a lot of personal data like name, photo, phone number, or location.
If this data is collected or used without permission, it breaks privacy. Ethical AI should protect personal data and use it only with the user's consent.

3. Lack of Transparency

Many AI systems do not explain how they make decisions. This is called the black box problem.
People may not trust AI if they do not understand its decisions. Ethical AI should be clear and explainable, especially in important areas like healthcare and banking.

4. Responsibility and Accountability

When AI makes a mistake, it is not clear who is responsible—the machine, the programmer, or the company.
Ethically, humans must take responsibility for AI decisions and errors.

5. Job Loss and Unemployment

AI and automation can replace human workers in many jobs.
This may cause unemployment and economic problems. Ethical use of AI should focus on helping humans learn new skills and creating new job opportunities.

6. Safety and Reliability

AI systems must work correctly and safely.
If AI fails in areas like medical diagnosis or self-driving cars, it can cause serious harm. Ethical AI must be tested properly to avoid danger.

7. Misuse of AI

AI can be misused to create fake videos (deepfakes), spread false information, or help in cybercrime.
Such misuse is dangerous for society. Ethical AI means using technology only for good purposes.

8. Loss of Human Control

If people depend too much on AI, they may stop using their own thinking and judgment.
Important decisions should always involve human control. AI should help humans, not replace them completely.

Conclusion

Ethical issues in AI remind us that technology must be used carefully. AI should be fair, safe, transparent, and human-friendly. When used ethically, AI can greatly benefit society without causing harm.

3-Marks Questions (Ethical Issues in AI)

I. What is meant by ethical issues in Artificial Intelligence?
II. Explain bias in AI with one example.
III. Why is data privacy important in AI systems?
IV. What is the black box problem in AI?
V. Explain accountability in AI systems.
VI. How can AI cause job loss?
VII. Why is safety important in AI applications?
VIII. What is misuse of AI? Give two examples.
IX. Why is human control necessary in AI?
X. Write any three ethical principles of AI.

SUBJECT: ARTIFICIAL INTELLIGENCE Class XII Semester 4

Question Pattern: [Short Answer Questions , Descriptive Questions] MARKS: 35

Unit	SHORT ANSWER TYPE QUESTIONS (2 marks)	DESCRIPTIVE TYPE QUESTIONS (3/4/5 marks)	TOTAL
4 : Unsupervised Learning	3X2=6 [3 OUT OF 6 QUESTIONS]	3X3=9 [3 OUT OF 6 QUESTIONS]	15
5: Artificial Neural Network	2X2=4 [2 OUT OF 4 QUESTIONS]	2X5=10 [2 OUT OF 4 QUESTIONS] 1X3=3 [1 OUT OF 2 QUESTIONS]	17
6: Ethics in AI	-	1X3=3 [1 OUT OF 2 QUESTIONS]	03
TOTAL	10	25	35

Question Structure to be followed :

Section-A [for 2 marks questions] 5X2=10

1. Answer any three questions out of six questions

I. What is unsupervised learning?
II. How is unsupervised learning different from supervised learning?
III. What is meant by clustering in machine learning?
IV. Why is clustering called an unsupervised learning technique?
V. Write two differences between supervised and unsupervised learning.
VI. Give two real-world applications of clustering.
VII. What is the main goal of clustering?
VIII. Define classification.
IX. Write two differences between clustering and classification.
X. What is K-means clustering algorithm?
XI. What is the role of distance measure in K-means clustering?
XII. What is meant by a cluster centroid?
XIII. Mention any two use cases of K-means clustering.
XIV. Why is K-means considered a simple clustering algorithm?
XV. State one advantage and one limitation of clustering.

2. Answer any two questions out of four questions

1. What is the biological motivation behind Artificial Neural Networks (ANN)?
2. What is the McCulloch and Pitts neuron model?

3. Write the mathematical expression of a simple artificial neuron.
4. What is an activation function? Name any two activation functions.
5. What is a threshold (step) activation function?
6. What is the sigmoid activation function? Write its equation.
7. Why is a perceptron called a linear classifier?
8. State the perceptron learning (training) rule.
9. How can the AND logic function be represented using a threshold perceptron?
10. What is meant by a linear separator in input space?
11. Why does a single-layer perceptron fail to solve the XOR problem?
12. What is the Delta rule used for in neural network training?
13. What is meant by overfitting in neural networks?
14. What is the role of a convolution layer in a Convolutional Neural Network (CNN)?
15. What is the function of a pooling layer in CNN?

Section-B [for 3 marks questions] 5X3=15

3. Answer any three questions out of six questions

1. Describe the steps involved in a clustering process.
2. Explain how similarity or distance affects clustering results.
3. What are the limitations of unsupervised learning techniques?
4. Explain why labeled data is not required in clustering.
5. Describe the working steps of the K-means algorithm.
6. What problems can occur if the value of K is chosen incorrectly in K-means?
7. Explain the effect of initial centroid selection in K-means clustering.
8. What is meant by intra-cluster distance and inter-cluster distance?
9. Explain how clustering is useful in data analysis and pattern discovery.
10. Suppose data points naturally form three groups. What value of K should be chosen in K-means and why?
11. If two data points are very close to each other, how will K-means treat them? Explain.
12. What will happen in K-means if all initial centroids are chosen very close to each other?
13. Given a small dataset, explain how clusters are updated after one iteration of K-means.
14. Why does K-means fail to detect non-spherical clusters? Explain with reason.
15. Explain the main objectives of unsupervised learning.

4. Answer any one question out of two questions

A. Explain how a biological neuron is mapped to an artificial neuron in ANN.
B. Compare threshold activation function and sigmoid activation function.

C. Explain the representational power of a single-layer perceptron.
D. Derive the equation of a linear decision boundary for a perceptron with two inputs.
E. Explain the Delta learning rule for training an unthresholded perceptron.
F. Why are hidden layers required in neural networks? Explain using the XOR problem.
G. Why is non-linearity important in neural networks?
H. Differentiate between feed-forward neural networks and recurrent neural networks.
I. Explain generalization and stopping criteria in neural network training.
J. Explain why CNNs are better than conventional feed-forward neural networks for image recognition tasks.

5. Answer any one question out of two questions
What is meant by ethical issues in Artificial Intelligence?
Explain bias in AI with one example.
Why is data privacy important in AI systems?
What is the black box problem in AI?
Explain accountability in AI systems.
How can AI cause job loss?
Why is safety important in AI applications?
What is misuse of AI? Give two examples.
Why is human control necessary in AI?
Write any three ethical principles of AI.

Section-C [for 5 marks questions] 2X5=10
6. Answer any two questions out of four questions

A perceptron has two inputs x1,x2with weights w1==1 and bias b=−1.5.
(a) Write the net input equation.
(b) Find the output for input (1,1) using a threshold activation function.
(c) Identify the logic function implemented.

3. Consider the following training example for an unthresholded perceptron:
Input x=(1,2), target output t=1, learning rate η=0.1 initial weights w=(0.2,0.1).
(a) Compute the actual output.
(b) Update the weights using the Delta rule.

4. A dataset is not linearly separable.
(a) Explain why a single-layer perceptron will fail.
(b) State how adding hidden layers solves this problem with reference to XOR.

5. Given a 4×4 input image and a 2×2 filter:
   (a) Explain how one step of 2-D convolution is performed.
   (b) State the size of the output feature map (stride = 1, no padding).

6. During training of a neural network, training accuracy is high but validation accuracy starts decreasing.
(a) Identify the problem.
(b) Suggest two methods to handle this problem.

7. A CNN contains a convolution layer followed by a pooling layer.
   (a) Explain the role of each layer in feature extraction.
   (b) State one advantage of pooling.

8. For handwritten digit recognition, the input image size is 28×28 pixels.
   (a) How many input neurons are required in a simple ANN?
   (b) How are output neurons represented for 10 digits (0–9)?
9. Explain the training of a perceptron as a linear classifier with a suitable example.
10. Describe the Backpropagation algorithm used for training multilayer feed-forward neural networks. Explain its main steps and learning process.
11. Explain the problem of overfitting in neural networks. How can validation data and early stopping be used to overcome overfitting?
12. With a neat block diagram, explain how a Convolutional Neural Network (CNN) is used for handwritten digit recognition.