GATE in Data Science and AI Study Materials
Machine Learning
By Piyush Wairale

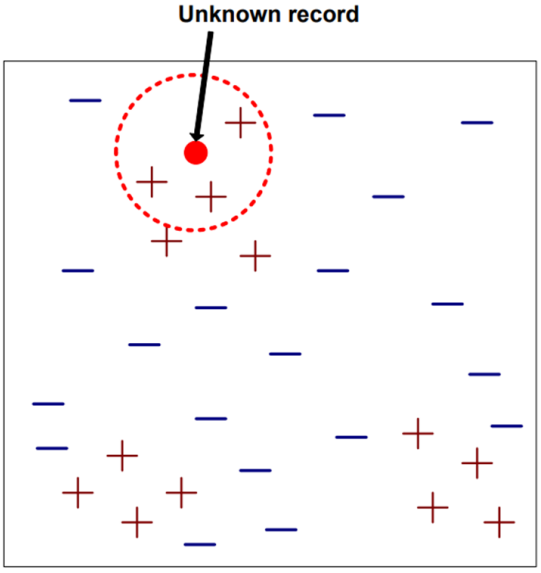
Instructions:

- Kindly go through the lectures/videos on our website www.piyushwairale.com
- Read this study material carefully and make your own handwritten short notes. (Short notes must not be more than 5-6 pages)
- Attempt the question available on portal.
- Revise this material at least 5 times and once you have prepared your short notes, then revise your short notes twice a week
- **If you are not able to understand any topic or required detailed explanation, please mention it in our discussion forum on webiste**
- **Let me know, if there are any typos or mistake in study materials. Mail me at piyushwairale100@gmail.com**

1 K-Nearest Neighbors

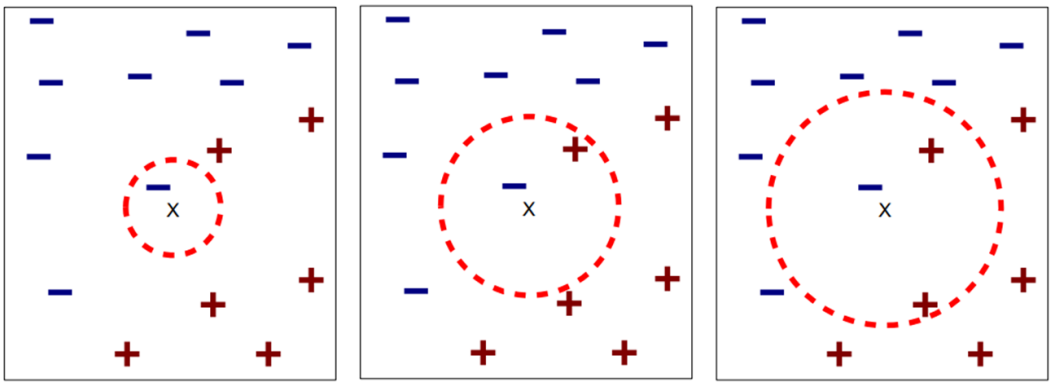
- K-Nearest Neighbors (KNN) is a simple and intuitive machine-learning algorithm used for both classification and regression tasks.
- It is a non-parametric and instance-based learning method, which means it doesn't make any assumptions about the underlying data distribution and makes predictions based on the similarity of data points.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.
- It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN tries to predict the correct class for the test data by calculating the distance between the test data and all the training points. Then select the K number of points which is closest to the test data. The KNN algorithm calculates the probability of the test data belonging to the classes of 'K' training data and class that holds the highest probability will be selected. In the case of regression, the value is the mean of the 'K' selected training points.

Nearest-Neighbor Classifiers



- Requires three things
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of *k*, the number of nearest neighbors to retrieve
- To classify an unknown record:
  - Compute distance to other training records
  - Identify *k* nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

Definition of Nearest Neighbor

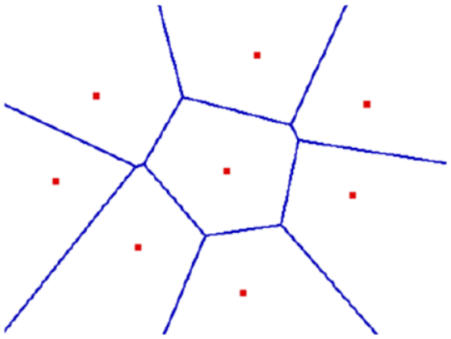


(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

K-nearest neighbors of a record x are data points that have the k smallest distance to x

Voronoi diagram, deals with when value of K =1

Describes the areas that are nearest to any given point, given a set of data. Each line segment is equidistant between two points of opposite class



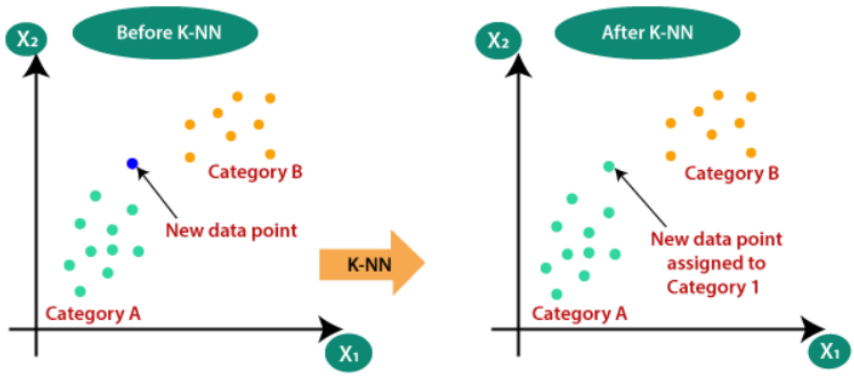
Choosing Value of K

- Larger k may lead to better performance But if we set k too large we may end up looking at samples that are not neighbors (are far away from the query)
- We can use cross-validation to find k
- Rule of thumb is k ¡ sqrt(n), where n is the number of training example
- Larger k produces smoother boundary effect
- When K==N, always predict the majority class

1.1 Working

The K-NN working can be explained on the basis of the below algorithm:

1. Select the number K of the neighbors
2. Calculate the Euclidean distance of K number of neighbors
3. Take the K nearest neighbors as per the calculated Euclidean distance.
4. Among these k neighbors, count the number of the data points in each category.
5. Assign the new data points to that category for which the number of the neighbor is maximum.
6. Our model is ready.



1.2 K-Nearest Neighbors (KNN) Classification Example

Suppose we have a dataset with the following points:

Data Point	Feature 1 (X1)	Feature 2 (X2)	Class
A	1	2	Blue
B	2	3	Blue
C	2	1	Red
D	3	3	Red
E	4	2	Blue

Now, let's say we want to classify a new data point with features $X1 = 2.5$ and $X2 = 2.5$ using a KNN algorithm with $k = 3$ (i.e., considering the three nearest neighbors).

1. Calculate Euclidean Distances:

$$\text{Distance to A: } \sqrt{(2.5-1)^2 + (2.5-2)^2} = \sqrt{1.5}$$
$$\text{Distance to B: } \sqrt{(2.5-2)^2 + (2.5-3)^2} = \sqrt{1.5}$$
$$\text{Distance to C: } \sqrt{(2.5-2)^2 + (2.5-1)^2} = \sqrt{1.5}$$
$$\text{Distance to D: } \sqrt{(2.5-3)^2 + (2.5-3)^2} = \sqrt{2.5}$$
$$\text{Distance to E: } \sqrt{(2.5-4)^2 + (2.5-2)^2} = \sqrt{4.5}$$

2. Find K Nearest Neighbors: Identify the three nearest neighbors based on the calculated distances. In this case, the three closest points are A, B, and C.

3. Majority Voting: Determine the majority class among the three nearest neighbors. Since A and B are Blue, and C is Red, the majority class is Blue.

4. Prediction: Predict that the new point $X1 = 2.5, X2 = 2.5$ belongs to the majority class, which is Blue.

1.3 Advantages of K-Nearest Neighbors (KNN):

- **Simplicity:** KNN is easy to understand and implement, making it a good choice for simple classification and regression tasks.
- **No Training Period:** KNN is a lazy learner, meaning it doesn't require a lengthy training period. It stores the entire training dataset and makes predictions when needed.
- **Non-parametric:** KNN doesn't make any assumptions about the underlying data distribution, making it versatile for a wide range of applications.
- **Adaptability:** KNN can be used for both classification and regression tasks, and it can handle multi-class problems without modification.
- **Interpretability:** The algorithm provides human-interpretable results, as predictions are based on the majority class or the average of the nearest neighbors.

1.4 Disadvantages of K-Nearest Neighbors (KNN):

- **Computational Intensity:** KNN can be computationally expensive, especially for large datasets, as it requires calculating distances to all data points during prediction.
- **Sensitivity to Feature Scaling:** KNN is sensitive to the scale of features, so it's essential to normalize or standardize your data before applying the algorithm.
- **Curse of Dimensionality:** KNN's performance degrades as the number of features or dimensions increases, as distances between data points become less meaningful in high-dimensional spaces.
- **Determining the Optimal K:** Selecting the right value for K is crucial, and choosing an inappropriate K can lead to underfitting or overfitting. There's no universally optimal value, and it often requires experimentation.
- **Imbalanced Data:** KNN can be biased towards the majority class in imbalanced datasets. It's essential to balance the dataset or adjust the class weights when necessary.

2 Naive Bayes Classifier

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.

2.1 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: Where,

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

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.

Assumption

The fundamental Naive Bayes assumption is that each feature makes an **independent** and **equal** contribution to the outcome.

With relation to our dataset, this concept can be understood as:

We assume that no pair of features are dependent. For example, the temperature being 'Hot' has nothing to do with the humidity or the outlook being 'Rainy' has no effect on the winds. Hence, the features are assumed to be independent.

Secondly, each feature is given the same weight(or importance). For example, knowing only temperature and humidity alone can't predict the outcome accurately. None of the attributes is irrelevant and assumed to be contributing equally to the outcome.

Here are the key concepts and characteristics of the Naive Bayes classifier:

- **Bayes' Theorem:** The classifier is based on Bayes' theorem, which calculates the probability of a hypothesis (in this case, a class label) given the evidence (features or attributes). Mathematically, it is expressed as P(class—evidence) = [P(evidence—class) * P(class)] / P(evidence).

- **Independence Assumption:** The "Naive" in Naive Bayes refers to the assumption that all features are independent of each other, given the class label. In reality, this assumption is often not true, but the simplification makes the algorithm computationally efficient and easy to implement.
- **Types of Naive Bayes Classifiers:**
  1. Multinomial Naive Bayes: Typically used for text classification where features represent word counts.
  2. Gaussian Naive Bayes: Suitable for continuous data and assumes a Gaussian distribution of features.
  3. Bernoulli Naive Bayes: Applicable when features are binary, such as presence or absence.
- **Prior Probability (P(class)):** This is the probability of a class occurring before observing any evidence. It can be calculated from the training data.
- **Likelihood (P(evidence—class)):** This represents the probability of observing a specific set of features given a class label. For text classification, this might involve counting the occurrence of words in documents.
- **Posterior Probability (P(class—evidence)):** It is the probability of a class label given the evidence, which is what the Naive Bayes classifier calculates for each class.
- **Classification:** To classify a new data point, the classifier calculates the posterior probabilities for each class and selects the class with the highest probability.

Bayes Classifiers

- Bayesian classifiers use **Bayes theorem**, which says

$$p(c_j \mid d) = \frac{p(d \mid c_j)\, p(c_j)}{p(d)}$$

- $p(c_j \mid d)$ = probability of instance d being in class c_j,
  This is what we are trying to compute
- $p(d \mid c_j)$ = probability of generating instance d given class c_j,
  We can imagine that being in class c_j, causes you to have feature d with some probability
- $p(c_j)$ = probability of occurrence of class c_j,
  This is just how frequent the class c_j, is in our database
- $p(d)$ = probability of instance d occurring

  This can actually be ignored, since it is the same for all classes

- To simplify the task, **naïve Bayesian classifiers** assume attributes have independent distributions, and thereby estimate

$$p(d|c_j) = p(d_1|c_j) * p(d_2|c_j) * \ldots * p(d_n|c_j)$$

The probability of class c_j generating instance d, equals….

The probability of class c_j generating the observed value for feature 1, multiplied by..

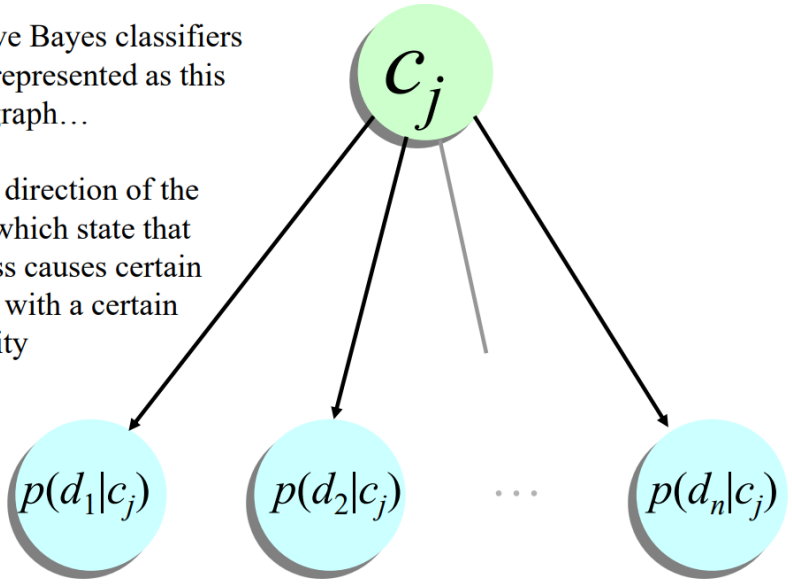
The probability of class c_j generating the observed value for feature 2, multiplied by..

- To simplify the task, **naïve Bayesian classifiers** assume attributes have independent distributions, and thereby estimate

$$p(d|c_j) = p(d_1|c_j) * p(d_2|c_j) * \ldots * p(d_n|c_j)$$

The Naive Bayes classifiers is often represented as this type of graph…

Note the direction of the arrows, which state that each class causes certain features, with a certain probability

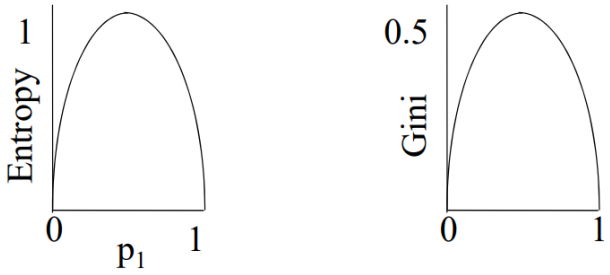


- Approach:
  - compute the posterior probability P(C | A_1, A_2, …, A_n) for all values of C using the Bayes theorem

$$P(C \mid A_1 A_2 \ldots A_n) = \frac{P(A_1 A_2 \ldots A_n \mid C) P(C)}{P(A_1 A_2 \ldots A_n)}$$

  - Choose value of C that maximizes
    P(C | A_1, A_2, …, A_n)
  - Equivalent to choosing value of C that maximizes
    P(A_1, A_2, …, A_n|C) P(C)

- How to estimate P(A_1, A_2, …, A_n | C)?



- Information gain on partitioning S into r subsets
- Impurity (S) - sum of weighted impurity of each subset

$$Gain(S, S_1..S_r) = Entropy(S) - \sum_{j=1}^{r} \frac{S_j}{S} Entropy(S_j)$$

2.2 Advantages of Naive Bayes:

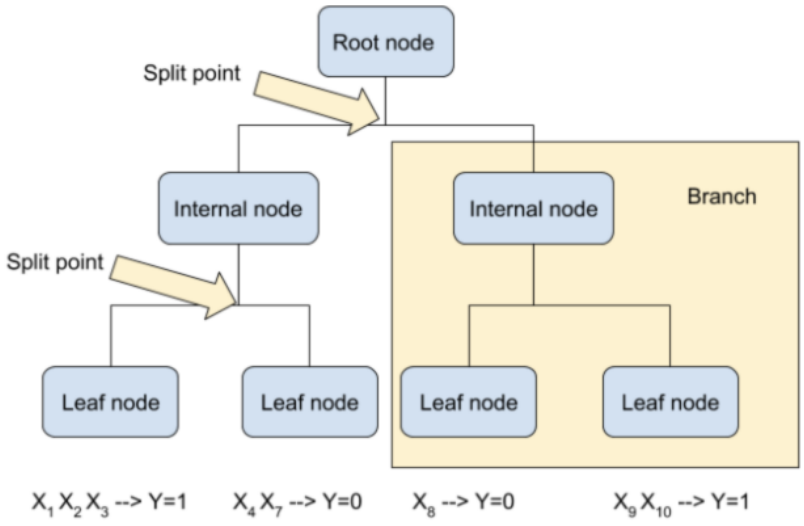
- Simplicity: Naive Bayes is easy to implement and understand, making it a good choice for quick classification tasks.
- Efficiency: It can handle a large number of features efficiently, particularly in text classification.
- Works Well with Small Datasets: It can perform reasonably well even with limited training data.
- Multiclass Classification: It can be used for multiclass classification problems.
- Interpretable: The results are easy to interpret, as it provide the probability of belonging to each class.

2.3 Disadvantages of Naive Bayes:

- **Independence Assumption:** The assumption of feature independence doesn't always hold, which can affect accuracy.
- Sensitivity to Feature Distribution: It may not perform well when features have complex, non-Gaussian distributions.
- Requires Sufficient Data: For some cases, Naive Bayes might not perform well when there is a scarcity of data.
- Zero Probability Problem: If a feature-class combination does not exist in the training data, the probability will be zero, causing issues. Smoothing techniques are often used to address this.

3 Decision Trees

- A decision tree is a simple model for supervised classification. It is used for classifying a single discrete target feature.
- Each internal node performs a Boolean test on an input feature (in general, a test may have more than two options, but these can be converted to a series of Boolean tests). The edges are labeled with the values of that input feature.
- Each leaf node specifies a value for the target feature
- Classifying an example using a decision tree is very intuitive. We traverse down the tree, evaluating each test and following the corresponding edge. When a leaf is reached, we return the classification on that leaf.
- Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems.
- It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.
- In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset. It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.
- A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.



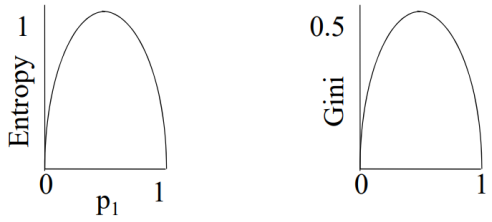
3.1 Terminologies

- **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 generalization and prevent overfitting.

3.2 Measures of impurity

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

1. Information Gain



- Information gain on partitioning S into r subsets
- Impurity (S) - sum of weighted impurity of each subset

$$Gain(S, S_1..S_r) = Entropy(S) - \sum_{j=1}^{r} \frac{S_j}{S} Entropy(S_j)$$

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

It can be calculated using the below formula:
Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)
Entropy: Entropy is a metric to measure the impurity in a given attribute.

$$Entropy \quad (S) = -\sum_{i=1}^{k} p_i \log \; p_i$$

It specifies randomness in data. Entropy can be calculated as:

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

2. Gini Index

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

Gini index can be calculated using the below formula:

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

3.3 Advantages of Decision Trees:

- Interpretability: Decision Trees are easy to interpret, making them a good choice when you need to explain or visualize the model's decisions.
- Handling Non-linearity: Decision Trees can capture non-linear relationships between features and the target variable.
- Feature Selection: They can automatically select the most important features, reducing the need for feature engineering.
- Versatility: Decision Trees can handle both categorical and numerical data.
- Efficiency: They are relatively efficient during prediction, with time complexity logarithmic in the number of data points.

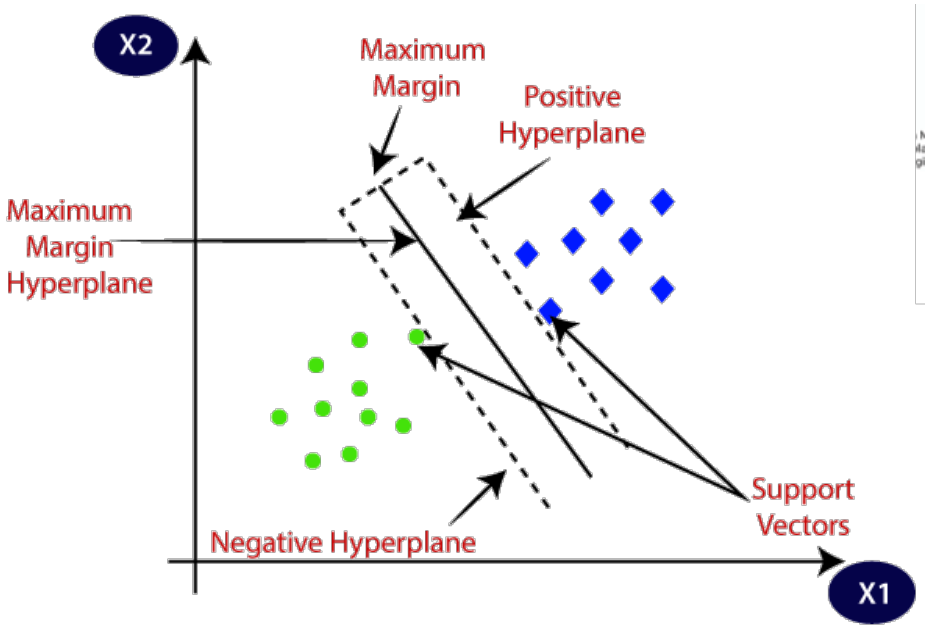
3.4 Disadvantages of Decision Trees:

- Overfitting: Decision Trees can be prone to overfitting, creating complex models that don't generalize well to new data. Pruning and setting appropriate parameters can help mitigate this.
- Bias Toward Dominant Classes: In classification tasks, Decision Trees can be biased toward dominant classes, leading to imbalanced predictions.
- Instability: Small variations in the data can lead to different tree structures, making them unstable models.
- Greedy Algorithm: Decision Trees use a greedy algorithm, making locally optimal decisions at each node, which may not lead to the global optimal tree structure

4 Support Vector Machine

- Support Vector Machine is a system for efficiently training linear learning machines in kernel-induced feature spaces, while respecting the insights of generalisation theory and exploiting optimisation theory.'
- The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.
- SVMs pick best separating hyperplane according to some criterion e.g. maximum margin
- Training process is an optimisation
- Training set is effectively reduced to a relatively small number of support vectors
- SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

  Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



Here are the key concepts and characteristics of Support Vector Machines:

- **Hyperplane:** There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.
- In a binary classification problem, an SVM finds a hyperplane that best separates the data points of different classes. This hyperplane is the decision boundary.
- The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane.

  We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.
- **Support Vectors:**The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector. They are critical for defining the margin and determining the location of the hyperplane.
- **Margin:** The margin is the distance between the support vectors and the decision boundary. SVM aims to maximize this margin because a larger margin often leads to better generalization.
- **C Parameter:** The regularization parameter "C" controls the trade-off between maximizing the margin and minimizing the classification error. A smaller "C" value results in a larger margin but may allow some misclassifications, while a larger "C" value allows for fewer misclassifications but a smaller margin.
- **Multi-Class Classification:** SVMs are inherently binary classifiers, but they can be extended to handle multi-class classification using techniques like one-vs-one (OvO) or one-vs-all (OvA) classification.
- **The Scalar Product:**The scalar or dot product is, in some sense, a measure of Similarity $a.b = |a|.|b|cos(\theta)$
- **Decision Function for binary classification**

$$f(x) \in \mathbf{R}$$

$$f(x_i) \geq 0 \Rightarrow y_i = 1$$
$$f(x_i) < 0 \Rightarrow y_i = -1$$

- **Feature Spaces** We may separate data by mapping to a higherdimensional feature space
  – The feature space may even have an infinite number of dimensions!
  • We need not explicitly construct the new feature space

4.1 Kernels

We may use Kernel functions to implicitly map to a new feature space
• Kernel fn: $K(x_1, x_2) \in R$
• Kernel must be equivalent to an inner product in some feature space

- **Kernel Trick:** SVM can handle non-linearly separable data by using a kernel function to map the data into a higher-dimensional space where it becomes linearly separable. Common kernel functions include linear, polynomial, radial basis function (RBF), and sigmoid kernels.

- The linear classifier relies on inner product between vectors $K(\mathbf{x_i},\mathbf{x_j})=\mathbf{x_i}^T\mathbf{x_j}$
- If every datapoint is mapped into high-dimensional space via some transformation Φ: $\mathbf{x} \rightarrow \varphi(\mathbf{x})$, the inner product becomes:

$$K(\mathbf{x_i},\mathbf{x_j})= \varphi(\mathbf{x_i})^T\varphi(\mathbf{x_j})$$

- A *kernel function* is some function that corresponds to an inner product into some feature space.
- Example:

  2-dimensional vectors $\mathbf{x}=[x_1\ x_2]$; let $K(\mathbf{x_i},\mathbf{x_j})=(1 + \mathbf{x_i}^T\mathbf{x_j})^2$,

  Need to show that $K(\mathbf{x_i},\mathbf{x_j})= \varphi(\mathbf{x_i})^T\varphi(\mathbf{x_j})$:

$$K(\mathbf{x_i},\mathbf{x_j})=(1 + \mathbf{x_i}^T\mathbf{x_j})^2 = 1+ x_{i1}^2x_{j1}^2 + 2\,x_{i1}x_{j1}\,x_{i2}x_{j2}+ x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}=$$
$$= [1\ \ x_{i1}^2\ \sqrt{2}\,x_{i1}x_{i2}\ \ x_{i2}^2\ \sqrt{2}x_{i1}\ \sqrt{2}x_{i2}]^T\,[1\ \ x_{j1}^2\ \sqrt{2}\,x_{j1}x_{j2}\ \ x_{j2}^2\ \sqrt{2}x_{j1}\ \sqrt{2}x_{j2}] =$$
$$= \varphi(\mathbf{x_i})^T\varphi(\mathbf{x_j}), \quad \text{where } \varphi(\mathbf{x}) = [1\ \ x_1^2\ \sqrt{2}\,x_1x_2\ \ x_2^2\ \ \sqrt{2}x_1\ \sqrt{2}x_2]$$

- For some functions $K(\mathbf{x_i},\mathbf{x_j})$ checking that $K(\mathbf{x_i},\mathbf{x_j})= \varphi(\mathbf{x_i})^T\varphi(\mathbf{x_j})$ can be cumbersome.
- Mercer's theorem:

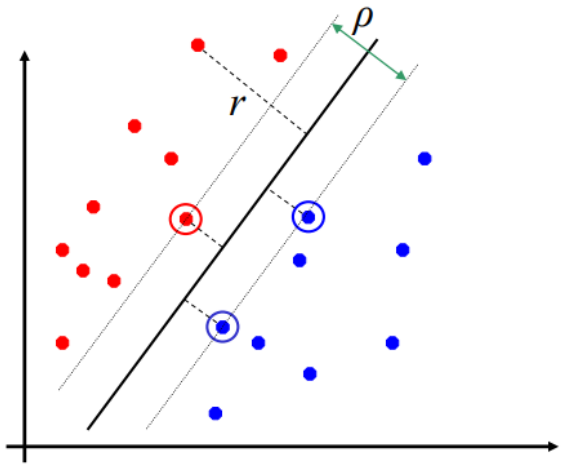
  Every semi-positive definite symmetric function is a kernel

Examples of kernel functions

- Linear: $K(\mathbf{x}_i,\mathbf{x}_j)= \mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j$
- Polynomial of power p: $K(\mathbf{x}_i,\mathbf{x}_j)= (1+ \mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j)^p$
- Gaussian (radial-basis function network): $K(\mathbf{x}_i,\mathbf{x}_j)= e^{-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{2\sigma^2}}$
- Two-layer perceptron: $K(\mathbf{x}_i,\mathbf{x}_j)= \tanh(\beta_0\mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j + \beta_1)$

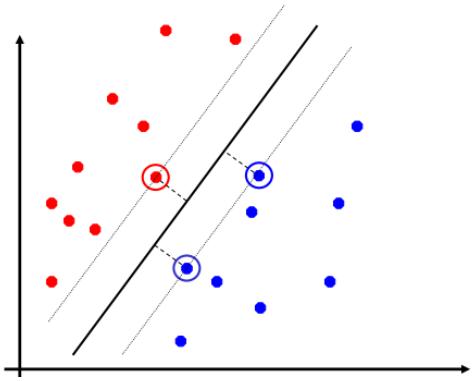
4.2 Classification Margin

- Distance from example data to the separator is $r = \frac{\mathbf{w}^T\mathbf{x}+b}{\|\mathbf{w}\|}$
- Data closest to the hyperplane are ***support vectors***.
- ***Margin*** ρ of the separator is the width of separation between classes.



Maximum Margin Classification

- Maximizing the margin is good according to intuition and theory.
- Implies that only support vectors are important; other training examples are ignorable.



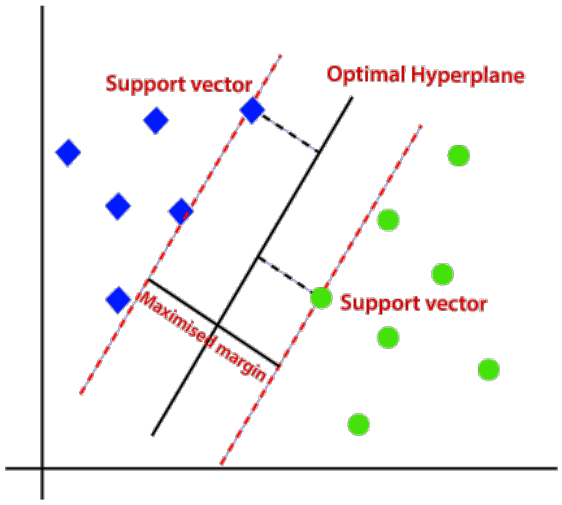
- Misclassification error and the function complexity bound generalization error.
- Maximizing margins minimizes complexity.
- "Eliminates" overfitting.
- Solution depends only on *Support Vectors* not number of attributes.

4.3 Types of SVM

1. Linear SVM: Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

   The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue. So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes.
   Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a hyperplane. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as margin. And the goal of SVM

is to maximize this margin. The hyperplane with the maximum margin is called the optimal hyperplane.



Linear SVM Mathematically

- Assuming all data is at distance larger than 1 from the hyperplane, the following two constraints follow for a training set $\{(\mathbf{x_i}, y_i)\}$

$$\mathbf{w}^T\mathbf{x_i} + b \geq 1 \quad \text{if } y_i = 1$$
$$\mathbf{w}^T\mathbf{x_i} + b \leq -1 \quad \text{if } y_i = -1$$

- For support vectors, the inequality becomes an equality; then, since each example's distance from the
- hyperplane is $r = \frac{\mathbf{w}^T\mathbf{x} + b}{\|\mathbf{w}\|}$ the margin is: $\rho = \frac{2}{\|\mathbf{w}\|}$

Linear SVMs Mathematically (cont.)

- Then we can formulate the *quadratic optimization problem:*

Find $\mathbf{w}$ and b such that

$\rho = \frac{2}{\|\mathbf{w}\|}$ is maximized and for all $\{(\mathbf{x_i}, y_i)\}$

$\mathbf{w}^T\mathbf{x_i} + b \geq 1$ if y_i=1; $\mathbf{w}^T\mathbf{x_i} + b \leq -1$ if $y_i = -1$

A better formulation:

Find $\mathbf{w}$ and b such that

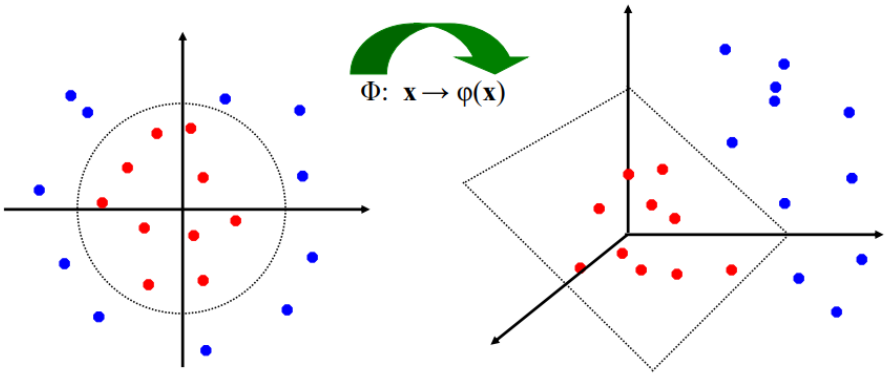
$\Phi(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w}$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$

$y_i(\mathbf{w}^T\mathbf{x_i} + b) \geq 1$

2. Non-linear SVM: Non-linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

Non-linear SVMs: Feature spaces

- General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable:



4.4 Advantages of Support Vector Machines:

- Effective in High-Dimensional Spaces: SVMs perform well even in high-dimensional feature spaces.
- Robust to Overfitting: SVMs are less prone to overfitting, especially when the margin is maximized. Accurate for Non-Linear Data: The kernel trick allows SVMs to work effectively on non-linear data by transforming it into higher dimensions.
- Wide Applicability: SVMs can be applied to various tasks, including classification, regression, and outlier detection.
- Strong Theoretical Foundation: SVMs are based on solid mathematical principles.

4.5 Disadvantages of Support Vector Machines:

- Computationally Intensive: Training an SVM can be computationally expensive, especially for large datasets.
- Sensitivity to Kernel Choice: The choice of the kernel function and kernel parameters can significantly impact the SVM's performance.
- Challenging for Large Datasets: SVMs may not be suitable for very large datasets because of their computational complexity.
- Interpretability: The decision boundary learned by SVMs can be challenging to interpret, especially in high-dimensional spaces.