Artificial Intelligence & Machine Learning Unit 3: Classification & Regression Question bank and its solution

Presentation · March 2022

DOI: 10.13140/RG.2.2.29226.70080

1 author:

Abhishek D. Patange
ABB
115 PUBLICATIONS **753** CITATIONS

SEE PROFILE

Artificial Intelligence & Machine Learning

Course Code: 302049

Unit 3: Classification & Regression

Third Year Bachelor of Engineering (Choice Based Credit System)
Mechanical Engineering (2019 Course)
Board of Studies – Mechanical and Automobile Engineering, SPPU, Pune
(With Effect from Academic Year 2021-22)

Question bank and its solution

by

Abhishek D. Patange, Ph.D.
Department of Mechanical Engineering
College of Engineering Pune (COEP)

Unit 3: Classification & Regression

Syllabus:

Content	Theory	Mathematics	Numerical
• **Classification & Regression**			
Decision tree (C & R)	✓	✓	✓
Random forest (C & R)	✓	✓	✗
Naive Bayes (C)	✓	✓	✓
Support vector machine (C & R)	✓	✓	✓
Logistic Regression (R)	✓	✓	✗
K-Means, K-Nearest Neighbor (KNN)	✓	✓	✗
• **Applications of classification and regression algorithms in Mechanical Engineering**			

Note: 'C' stands for classification and 'R' stands for regression

Type of question and marks:

Type	Theory	Mathematics	Numerical
Marks	2 or 4 or 6	4 marks	2 or 4 marks

Topic: Decision trees

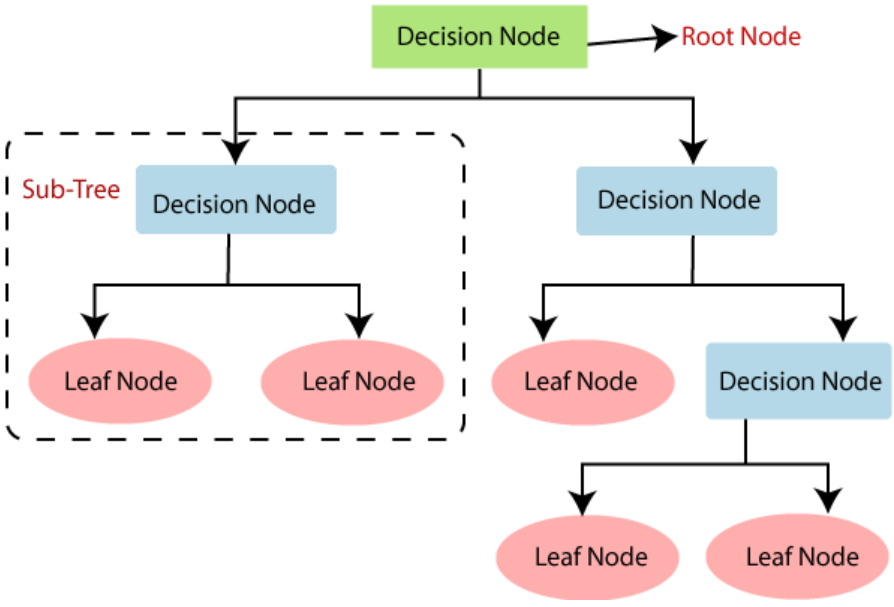
Theory	Mathematics	Numerical
✓	✓	✓

Theory questions

1. Why use decision trees?

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

- Decision 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.
- A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into sub-trees. Below diagram explains the general structure of a decision tree.



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

2. Explain decision tree terminology.

The decision tree comprises of root node, leaf node, branch nodes, parent/child node etc. following is the explanation of this terminology.

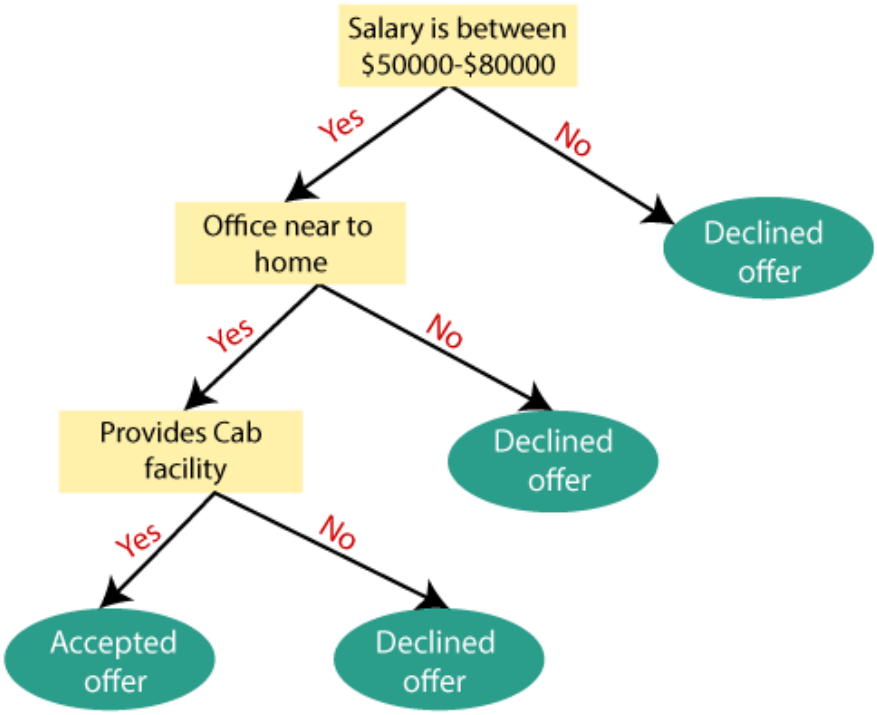
- **Root Node:** Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
- **Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
- **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
- **Branch/Sub Tree:** A tree formed by splitting the tree.
- **Pruning:** Pruning is the process of removing the unwanted branches from the tree.
- **Parent/Child node:** The root node of the tree is called the parent node, and other nodes are called the child nodes.

3. How does the Decision Tree algorithm Work for classification?

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node. For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree.

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

Example: Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM).



The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels. The next decision node further gets split into one decision node (Cab facility) and one leaf node. Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer). See the above figure.

4. How does the Decision Tree algorithm Work for regression?

The general idea is that we will segment the predictor space into a number of simple regions. In order to make a prediction for a given observation, we typically use the mean of the training data in the region to which it belongs. Since the set of splitting rules used to segment the predictor space can be summarized by a tree such approaches are called decision tree methods. These methods are simple and useful for interpretation. We want to predict a response or class Y from inputs $X_1, X_2, \ldots X_p$. We do this by growing a binary tree. At each internal node in the tree, we apply a test to one of the inputs, say X_i . Depending on the outcome of the test, we go to either the left or the right sub-branch of the tree. Eventually we come to a leaf node, where we make a prediction. This prediction aggregates or averages all the training data points which reach that leaf. In order to motivate regression trees, we begin with a simple example. Our motivation is to predict a baseball player's Salary based on Years (the number of years that he has played in the major leagues) and Hits (the number of hits that he made in the previous year). We first remove observations that are

missing Salary values and log-transform Salary so that its distribution has more of a typical bell-shape. Recall that Salary is measured in thousands of dollars.

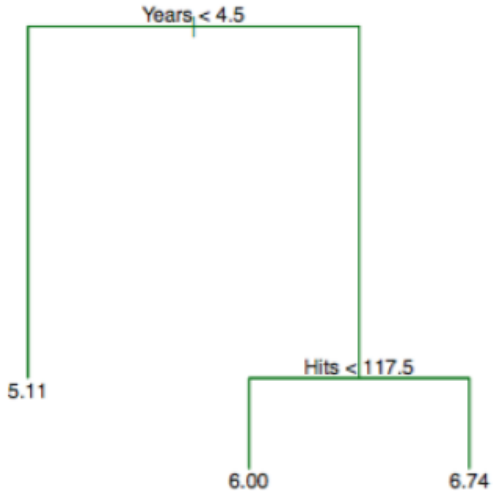


Figure 1: A regression tree for predicting the log salary of a baseball player, based on the number of years that he has played in the major leagues and the number of hits that he made in the previous year. At a given internal node, the label (of the form $X_j < t_k$) indicates the left-hand branch resulting from that split, and the right-hand branch corresponds to $X_j \geq t_k$.

The tree represents a series of splits starting at the top of the tree. The top split assigns observations having Years < 4.5 to the left branch. The predicted salary for these players is given by the mean response value for the players in the data set with Years < 4.5.For such players, the mean log salary is 5.107, and so we make a prediction of e5.107 thousands of dollars, i.e. 165, 174. How would you interpret the rest (right branch) of the tree?

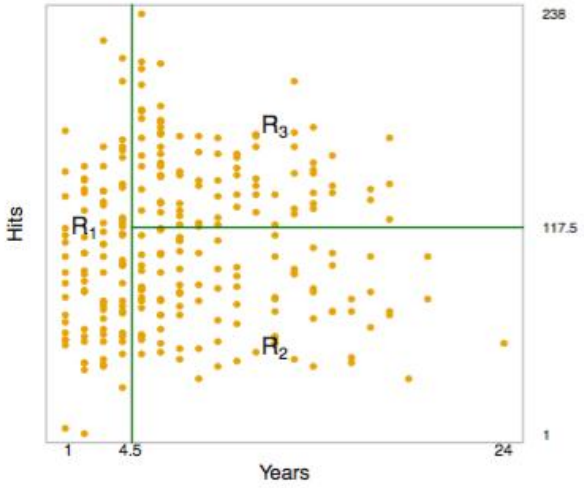


Figure 2: The three-region partition for the Hitters data set from the regression tree illustrated in Figure 2.

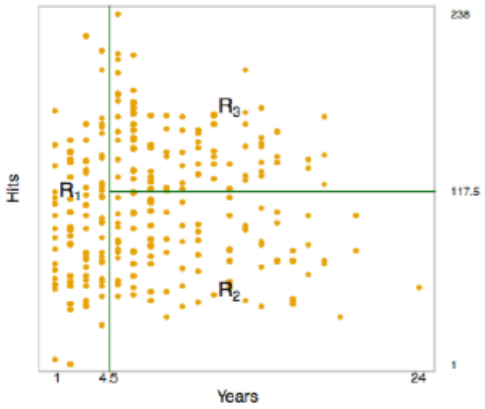


Figure 3: The three-region partition for the Hitters data set from the regression tree illustrated in Figure 2.

We can write these regions as the following:

1. $R_1 = X \mid Years < 4.5$
2. $R_2 = X \mid Years \geq 4.5, Hits < 117.5$
3. $R_3 = X \mid Years \geq 4.5, Hits \geq 117.5.$

- In keeping with the tree analogy, the regions *R*1, *R*2, and *R*3 are known as **terminal nodes** or **leaves** of the tree.
- As is the case for Figure 2, decision trees are typically drawn upside down, in the sense that the **leaves are at the bottom of the tree**.
- The points along the tree where the predictor space is split are referred to as **internal nodes**.
- In Figure 2, the two internal nodes are indicated by the text *Years* < 4.5 and *Hits* < 117.5.
- We refer to the segments of the trees that connect the nodes as **branches**.
- Years is the most important factor in determining Salary, and players with less experience earn lower salaries than more experienced players.
- Given that a player is less experienced, the number of hits that he made in the previous year seems to play little role in his salary.
- But among players who have been in the major leagues for five or more years, the number of hits made in the previous year does affect salary, and players who made more hits last year tend to have higher salaries.
- The regression tree shown in Figure 2 is likely an over-simplification of the true relationship between Hits, Years, and Salary, but it's a very nice easy interpretation over more complicated approaches.

1. We divide the predictor space—that is, the set of possible values for $X_1, \ldots, X_p$—into J distinct and non-overlapping regions, $R_1, \ldots, R_J$.
2. For every observation that falls into the region R_j, we make the same prediction, which is simply the mean of the response values for the training observations in R_j.

Suppose that in Step 1, we obtain two regions and that the response mean of the training observations in the first region is 10, while the response mean in the second region is 20. Then for a given observation $X = x$, if $x \in R_1$, we will predict a value of 10, and if $x \in R_2$, we will predict a value of 20.

But how do we actually construct the regions?

- The regions in theory could have any shape.
- However, we choose to divide the predictor space into high-dimensional rectangles or boxes (for simplicity and ease of interpretation of the resulting predictive model).

Our goal is to find boxes $R_1, \ldots, R_J$ that minimize the RSS given by

$$RSS = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2, \tag{1}$$

where $\hat{y}_{R_j}$ is the mean response for the training observations within the jth box.

- Computationally infeasible to consider every possible partition of the feature space into J boxes.
- Thus, we take a top-down, greedy approach called *recursive binary splitting*.
  - Called top-down since it begins at the top of the tree (all observations below to a single region) and then successively splits the predictor space.
  - Each split is indicated via two new branches further down on the tree.
  - It is greedy since at each step of the tree building process, the best split is made at that particular split (rather than looking ahead and picking a split that will lead to a better tree in a future split).

Mathematics based questions

5. Explain entropy reduction, information gain and Gini index in decision tree.

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

Information Gain:

- 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 Average) * Entropy (each feature)]

Entropy:

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

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

Where, S= Total number of samples, P(yes)= probability of yes, P(no)= probability of no

Gini Index:

- Gini index is a measure of impurity or purity used while creating a decision tree in the CART (Classification and Regression Tree) algorithm.
- An attribute with the low Gini index should be preferred as compared to the high Gini index. Gini index can be calculated using the formula: **Gini Index= $1 - \sum_j P_j^2$**
- It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.

6. What are advantages and limitations of the decision trees?

Advantages of the Decision Tree

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

Disadvantages of the Decision Tree

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

7. Many times while training decision tree tends to overfit. What is the reason behind it and how to avoid it?

Decision tree tends to overfit since at each node, it will make the decision among a subset of all the features (columns), so when it reaches a final decision, it is a complicated and long decision chain. Only if a data point satisfies all the rules along this chain, the final decision can be made. This kind of specific rule on training dataset make it very specific for the training set, on the other hand, cannot generalize well for new data points that it has never seen. Especially when your dataset has many features (high dimension), it tends to overfit more. In J48 decision tree, over fitting happens when algorithm gets information with exceptional attributes. This causes many fragmentations in the process distribution. Statistically unimportant nodes with least examples are known as fragmentations. Usually J48 algorithm builds trees and grows its branches 'just deep enough to perfectly classify the training examples'. This approach performs better with noise free data. But most of the time this strategy overfits the training examples with noisy data. At present there are two strategies which are widely used to bypass this overfitting in decision tree learning. Those are: 1) If tree grows taller, stop it from growing before it reaches the maximum point of accurate classification of the training data. 2) Let the tree to over-fit the training data then post-prune tree. By default, the decision tree model is allowed to grow to its full depth. Pruning refers to a technique to remove the parts of the decision tree to prevent growing to its full depth. By tuning the hyperparameters of the decision tree model one can prune the trees and prevent them from overfitting. There are two types of pruning Pre-pruning and Post-pruning. Now let's discuss the in-depth understanding and hands-on implementation of each of these pruning techniques.

Pre-Pruning:

The pre-pruning technique refers to the early stopping of the growth of the decision tree. The pre-pruning technique involves tuning the hyperparameters of the decision tree model prior to the training pipeline. The hyperparameters of the decision tree including **max_depth**, **min_samples_leaf**, **min_samples_split** can be tuned to early stop the growth of the tree and prevent the model from overfitting.

Post-Pruning:

The Post-pruning technique allows the decision tree model to grow to its full depth, then removes the tree branches to prevent the model from overfitting. ***Cost complexity pruning (ccp)*** is one type of post-pruning technique. In case of cost complexity pruning, the **ccp_alpha** can be tuned to get the best fit model.

Problems/Numerical

8. Problems on calculating entropy and information gain

Problem 1:

If we decided to arbitrarily label all 4 gumballs as red, how often would one of the gumballs is incorrectly labelled?

4 red and 0 blue:

$$Gini\ Index = 1-(probability\_red^2+probability\_blue^2) = 1-(1^2+0^2) = 0$$

The impurity measurement is 0 because we would never incorrectly label any of the 4 red gumballs here. If we arbitrarily chose to label all the balls 'blue', then our index would still be 0, because we would always incorrectly label the gumballs.

The gini score is always the same no matter what arbitrary class you take the probabilities of because they always add to 0 in the formula above.

A gini score of 0 is the most pure score possible.

2 red and 2 blue:

$$Gini\ Index = 1-(probability\_red^2+probability\_blue^2) = 1-(0.5^2+0.5^2) = 0.5$$

The impurity measurement is 0.5 because we would incorrectly label gumballs wrong about half the time. Because this index is used in binary target variables (0,1), a gini index of 0.5 is the least pure score possible. Half is one type and half is the other. **Dividing gini scores by 0.5 can help intuitively understand what the score represents. 0.5/0.5 = 1, meaning the grouping is as impure as possible (in a group with just 2 outcomes).**

3 red and 1 blue:

$$Gini\ Index = 1-(probability\_red^2+probability\_blue^2)=1-(0.75^2+0.25^2)=0.375$$

The impurity measurement here is 0.375. If we divide this by 0.5 for more intuitive understanding we will get 0.75, which is the probability of incorrectly/correctly labeling.

Problem 2:

How does entropy work with the same gumball scenarios stated in problem 1?

4 red and 0 blue:

$$Entropy = [(probability\_red) * log2(probability\_red)]—[(probability\_blue) * log2(probability\ of\ blue)] = [(4/4) * log2(4/4)] -[(0/4) * log2(0/4)] = 0$$

Unsurprisingly, the impurity measurement is 0 for entropy as well. This is the max purity score using information entropy.

2 red and 2 blue:

$$Entropy = [(probability\_red) * log2(probability\_red)] — [(probability\_blue) * log2(probability\ of\ blue)] = [(2/4) * log2(2/4)] - [(2/4) * log2(2/4)] = 1$$

The impurity measurement is 1 here, as it's the maximum impurity obtainable.

3 red and 1 blue:

$$Entropy = [(probability\_red) * log2(probability\_red)] — [(probability\_blue) * log2(probability\ of\ blue)] = [(3/4) * log2(3/4)] - [(1/4) * log2(1/4)] = 0.811$$

The purity/impurity measurement is 0.811 here, a bit worse than the gini score.

Problem 3:

Calculate entropy for following example.

For the set X = {a,a,a,b,b,b,b,b}

Total instances: 8

Instances of b: 5

Instances of a: 3

$$EntropyH(X) = -\left[\left(\frac{3}{8}\right) log_2 \frac{3}{8} + \left(\frac{5}{8}\right) log_2 \frac{5}{8}\right]$$

= - [0.375 * (-1.415) + 0.625 * (-0.678)]

= - (-0.53-0.424)

= 0.954

Problem 4:

In the below mini-dataset, the label we're trying to predict is the type of fruit. This is based off the size, color, and shape variables.

Fruit	Size	Color	Shape
Watermelon	Big	Green	Round
Apple	Medium	Red	Round
Banana	Medium	Yellow	Thin
Grape	Small	Green	Round
Grapefruit	Medium	Yellow	Round
Lemon	Small	Yellow	Round

Calculate the information gained if we select the *color* variable.

3 out of the 6 records are yellow, 2 are green, and 1 is red. Proportionally, the probability of a yellow fruit is 3 / 6 = 0.5; 2 / 6 = 0.333 for green, and 1 / 6 = 0.1666 for red. Using the formula from above, we can calculate it like this:

Information gain = - ([3/6 * $\log_2$(3/6)] + [2/6 * $\log_2$(2/6)] + [1/6 * $\log_2$(1/6)]) = **1.459148**

Calculate the information gained if we select the *size* variable.

Information gain = - ([3/6 * $\log_2$(3/6)] + [2/6 * $\log_2$(2/6)] + [1/6 * $\log_2$(1/6)]) = **1.459148**

In this case, 3 / 6 of the fruits are medium-sized, 2 / 6 are small, 1 / 6 is big.

Calculate the information gained if we select the *shape* variable.

Here, 5 / 6 of the fruits are round and 1 / 6 is thin.

Information gain = - ([5/6 * $\log_2$(5/6)] + [1/6 * $\log_2$(1/6)]) = **0.650022**

Problem 5:

Consider the following data set for a binary class problem.

A	B	Class Label
T	F	+
T	T	+
T	T	+
T	F	−
T	T	+
F	F	−
F	F	−
F	F	−
T	T	−
T	F	−

(a) Calculate the information gain when splitting on A and B. Which attribute would the decision tree induction algorithm choose?

Answer:

The contingency tables after splitting on attributes A and B are:

	$A=T$	$A=F$
+	4	0
−	3	3

	$B=T$	$B=F$
+	3	1
−	1	5

The overall entropy before splitting is:

$$E_{orig} = -0.4 \log 0.4 - 0.6 \log 0.6 = 0.9710$$

The information gain after splitting on A is:

$$\begin{aligned} E_{A=T} &= -\frac{4}{7}\log\frac{4}{7} - \frac{3}{7}\log\frac{3}{7} = 0.9852 \\ E_{A=F} &= -\frac{3}{3}\log\frac{3}{3} - \frac{0}{3}\log\frac{0}{3} = 0 \\ \Delta &= E_{orig} - 7/10 E_{A=T} - 3/10 E_{A=F} = 0.2813 \end{aligned}$$

The information gain after splitting on B is:

$$\begin{aligned} E_{B=T} &= -\frac{3}{4}\log\frac{3}{4} - \frac{1}{4}\log\frac{1}{4} = 0.8113 \\ E_{B=F} &= -\frac{1}{6}\log\frac{1}{6} - \frac{5}{6}\log\frac{5}{6} = 0.6500 \\ \Delta &= E_{orig} - 4/10 E_{B=T} - 6/10 E_{B=F} = 0.2565 \end{aligned}$$

Therefore, attribute A will be chosen to split the node.

(b) Calculate the gain in the Gini index when splitting on A and B. Which attribute would the decision tree induction algorithm choose?

Answer:

The overall gini before splitting is:

$$G_{orig} = 1 - 0.4^2 - 0.6^2 = 0.48$$

The gain in gini after splitting on A is:

$$\begin{aligned} G_{A=T} &= 1 - \left(\frac{4}{7}\right)^2 - \left(\frac{3}{7}\right)^2 = 0.4898 \\ G_{A=F} &= 1 = \left(\frac{3}{3}\right)^2 - \left(\frac{0}{3}\right)^2 = 0 \\ \Delta &= G_{orig} - 7/10 G_{A=T} - 3/10 G_{A=F} = 0.1371 \end{aligned}$$

The gain in gini after splitting on B is:

$$\begin{aligned} G_{B=T} &= 1 - \left(\frac{1}{4}\right)^2 - \left(\frac{3}{4}\right)^2 = 0.3750 \\ G_{B=F} &= 1 = \left(\frac{1}{6}\right)^2 - \left(\frac{5}{6}\right)^2 = 0.2778 \\ \Delta &= G_{orig} - 4/10 G_{B=T} - 6/10 G_{B=F} = 0.1633 \end{aligned}$$

Therefore, attribute B will be chosen to split the node.

Problem 6:

Consider the training examples shown in Table below for a binary classification problem.

Customer ID	Gender	Car Type	Shirt Size	Class
1	M	Family	Small	C0
2	M	Sports	Medium	C0
3	M	Sports	Medium	C0
4	M	Sports	Large	C0
5	M	Sports	Extra Large	C0
6	M	Sports	Extra Large	C0
7	F	Sports	Small	C0
8	F	Sports	Small	C0
9	F	Sports	Medium	C0
10	F	Luxury	Large	C0
11	M	Family	Large	C1
12	M	Family	Extra Large	C1
13	M	Family	Medium	C1
14	M	Luxury	Extra Large	C1
15	F	Luxury	Small	C1
16	F	Luxury	Small	C1
17	F	Luxury	Medium	C1
18	F	Luxury	Medium	C1
19	F	Luxury	Medium	C1
20	F	Luxury	Large	C1

(a) Compute the Gini index for the overall collection of training examples.

Answer:

Gini $= 1 - 2 \times 0.5^2 = 0.5$.

(b) Compute the Gini index for the `Customer ID` attribute.

Answer:

The gini for each `Customer ID` value is 0. Therefore, the overall gini for `Customer ID` is 0.

(c) Compute the Gini index for the `Gender` attribute.

Answer:

The gini for `Male` is $1 - 2 \times 0.5^2 = 0.5$. The gini for `Female` is also 0.5. Therefore, the overall gini for `Gender` is $0.5 \times 0.5 + 0.5 \times 0.5 = 0.5$.

(d) Compute the Gini index for the `Car Type` attribute using multiway split.

Answer:

The gini for `Family` car is 0.375, `Sports` car is 0, and `Luxury` car is 0.2188. The overall gini is 0.1625.

(e) Compute the Gini index for the `Shirt Size` attribute using multiway split.

Answer:

The gini for `Small` shirt size is 0.48, `Medium` shirt size is 0.4898, `Large` shirt size is 0.5, and `Extra Large` shirt size is 0.5. The overall gini for `Shirt Size` attribute is 0.4914.

(f) Which attribute is better, `Gender`, `Car Type`, or `Shirt Size`?

Answer:

`Car Type` because it has the lowest gini among the three attributes.

(g) Explain why `Customer ID` should not be used as the attribute test condition even though it has the lowest Gini.

Answer:

The attribute has no predictive power since new customers are assigned to new `Customer IDs`.

Problem 7:

Consider the training examples shown in Table below for a binary classification problem.

Instance	a_1	a_2	a_3	Target Class
1	T	T	1.0	+
2	T	T	6.0	+
3	T	F	5.0	−
4	F	F	4.0	+
5	F	T	7.0	−
6	F	T	3.0	−
7	F	F	8.0	−
8	T	F	7.0	+
9	F	T	5.0	−

(a) What is the entropy of this collection of training examples with respect to the positive class?

Answer:

There are four positive examples and five negative examples. Thus, $P(+) = 4/9$ and $P(-) = 5/9$. The entropy of the training examples is $-4/9 \log_2(4/9) - 5/9 \log_2(5/9) = 0.9911$.

(b) What are the information gains of a_1 and a_2 relative to these training examples?

Answer:

For attribute a_1, the corresponding counts and probabilities are:

a_1	+	-
T	3	1
F	1	4

The entropy for a_1 is

$$\frac{4}{9}\left[-(3/4)\log_2(3/4) - (1/4)\log_2(1/4)\right] + \frac{5}{9}\left[-(1/5)\log_2(1/5) - (4/5)\log_2(4/5)\right] = 0.7616.$$

Therefore, the information gain for a_1 is $0.9911 - 0.7616 = 0.2294$.

For attribute a_2, the corresponding counts and probabilities are:

a_2	+	-
T	2	3
F	2	2

The entropy for a_2 is

$$\frac{5}{9}\left[-(2/5)\log_2(2/5) - (3/5)\log_2(3/5)\right] + \frac{4}{9}\left[-(2/4)\log_2(2/4) - (2/4)\log_2(2/4)\right] = 0.9839.$$

Therefore, the information gain for a_2 is $0.9911 - 0.9839 = 0.0072$.

(c) For a_3, which is a continuous attribute, compute the information gain for every possible split.

Answer:

a_3	Class label	Split point	Entropy	Info Gain
1.0	+	2.0	0.8484	0.1427
3.0	-	3.5	0.9885	0.0026
4.0	+	4.5	0.9183	0.0728
5.0 5.0	- -	5.5	0.9839	0.0072
6.0	+	6.5	0.9728	0.0183
7.0 7.0	+ -	7.5	0.8889	0.1022

The best split for a_3 occurs at split point equals to 2.

(d) What is the best split (among a_1, a_2, and a_3) according to the information gain?

Answer:

According to information gain, a_1 produces the best split.

(e) What is the best split (between a_1 and a_2) according to the classification error rate?

Answer:

For attribute a_1: error rate $= 2/9$.

For attribute a_2: error rate $= 4/9$.

Therefore, according to error rate, a_1 produces the best split.

(f) What is the best split (between a_1 and a_2) according to the Gini index?

Answer:

For attribute a_1, the gini index is

$$\frac{4}{9}\left[1-(3/4)^2-(1/4)^2\right]+\frac{5}{9}\left[1-(1/5)^2-(4/5)^2\right]=0.3444.$$

For attribute a_2, the gini index is

$$\frac{5}{9}\left[1-(2/5)^2-(3/5)^2\right]+\frac{4}{9}\left[1-(2/4)^2-(2/4)^2\right]=0.4889.$$

Since the gini index for a_1 is smaller, it produces the better split.

Topic: Random forest tree

Theory	Mathematics	Numerical
✓	✓	✗

Theory questions

9. Why use random forest trees?

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** which is a process of *combining multiple classifiers to solve a complex problem and to improve the performance of the model.*

As the name suggests, ***"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."*** Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. **The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.**

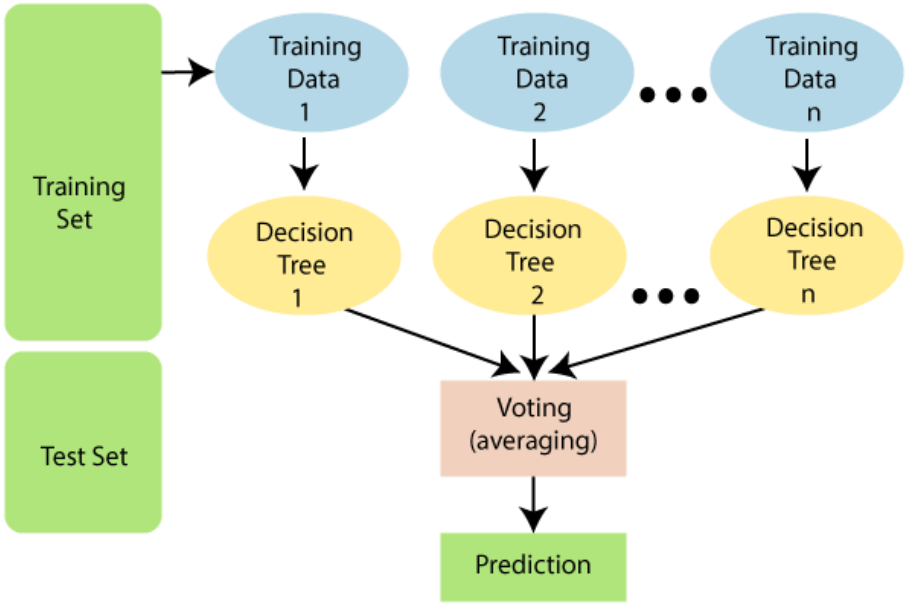
Assumptions for Random Forest

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But

together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random forest classifier:

- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations.

The below diagram explains the working of the Random Forest algorithm:



Below are some points that explain why we should use the Random Forest algorithm:

- It takes less training time as compared to other algorithms.
- It predicts output with high accuracy, even for the large dataset it runs efficiently.
- It can also maintain accuracy when a large proportion of data is missing.
- It can be used for both classifications as well as regression tasks.
- Overfitting problem that is censorious and can make results poor but in case of the random forest the classifier will not overfit if there are enough trees.
- It can be used for categorical values as well.
-

10. How does the random forest tree work for classification?

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

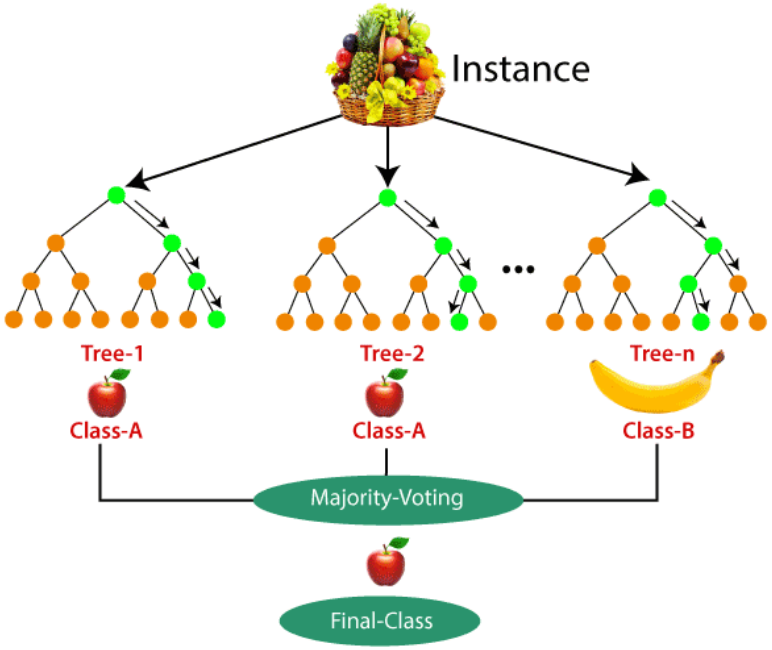
Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

The working of the algorithm can be better understood by the below example:

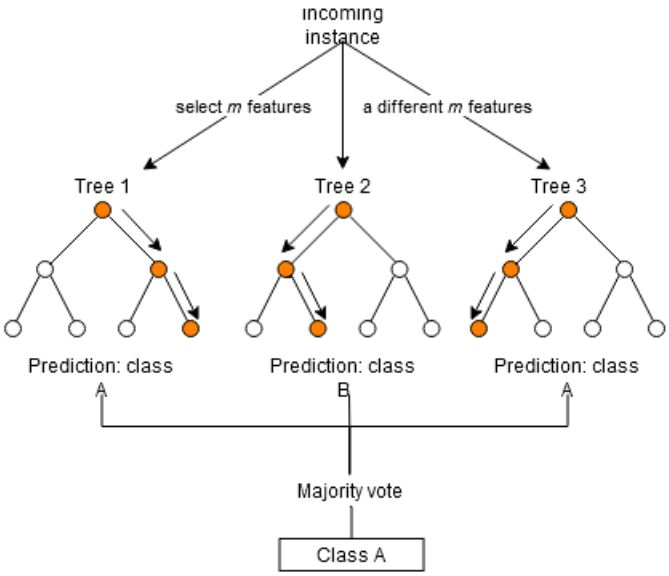
Example: Suppose there is a dataset that contains multiple fruit images. So, this dataset is given to the Random forest classifier. The dataset is divided into subsets and given to each decision tree. During the training phase, each decision tree produces a prediction result, and when a new data point occurs, then based on the majority of results, the Random Forest classifier predicts the final decision. Consider the below image:



11. Explain random forest tree terminology.

- **Bagging:** Given the training set of N examples, we repeatedly sample subsets of the training data of size n where n is less than N. Sampling is done at random but with replacement. This subsampling of a training set is called *bootstrap aggregating*, or *bagging*, for short.
- **Random subspace method:** If each training example has M features, we take a subset of them of size $m < M$ to train each estimator. So no estimator sees the full training set, each estimator sees only m features of n training examples.
- **Training estimators:** We create N_{tree} decision trees, or estimators, and train each one on a different set of m features and n training examples. The trees are not pruned, as they would be in the case of training a simple decision tree classifier.
- **Perform inference by aggregating predictions of estimators:** To make a prediction for a new incoming example, we pass the relevant features of this example to each of the N_{tree} estimators. We will obtain N_{tree} predictions, which we need to combine to produce the overall prediction of the random forest. In the case of classification, we will use

majority voting to decide on the predicted class, and in the case of regression, we will take the mean value of the predictions of all the estimators.



Random forest inference for a simple classification example with N_{tree} = 3

This use of many estimators is the reason why the random forest algorithm is called an *ensemble method*. Each individual estimator is a weak learner, but when many weak estimators are combined together they can produce a much stronger learner. Ensemble methods take a 'strength in numbers' approach, where the output of many small models is combined to produce a much more accurate and powerful prediction.

12. What are advantages and limitations of the random forest tree?

Advantages of Random Forest

- Random Forest is capable of performing both Classification and Regression tasks.
- It is capable of handling large datasets with high dimensionality.
- It enhances the accuracy of the model and prevents the overfitting issue.
- It is fast and can deal with missing values data as well.
- Using random forest you can compute the relative feature importance.
- It can give good accuracy even if the higher volume of data is missing.

Limitations of Random Forest

- Although random forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks.
- Random forest is a complex algorithm that is not easy to interpret.
- Complexity is large.
- Predictions given by random forest takes many times if we compare it to other algorithms
- Higher computational resources are required to use a random forest algorithm.

13. What is the difference between simple decision tree and random forest tree?

SN	Random Forest	Decision Tree
1.	While building a random forest the number of rows is selected randomly.	Whereas, it built several decision trees and find out the output.
2.	It combines two or more decision trees together.	Whereas the decision is a collection of variables or data set or attributes.
3.	It gives accurate results.	Whereas it gives less accurate results.
4.	By using multiple trees it reduces the chances of overfitting.	On the other hand, decision trees, it has the possibility of overfitting, which is an error that occurs due to variance or due to bias.
5.	Random forest is more complicated to interpret.	Whereas, the decision tree is simple so it is easy to read and understand.
6.	In a random forest, we need to generate, process, and analyze trees so that this process is slow, it may take one hour or even days.	The decision tree is not accurate but it processes fast which means it is fast to implement.
7.	It has more computation because it has n number of decision trees, so more decision trees more computation.	Whereas it has less computation.
8.	It has complex visualization, but it plays an important role to show hidden patterns behind the data.	On the other hand, it is simple to visualize because we just need to fit the decision tree model.
9.	The classification and regression problems can be solved by using random forest.	Whereas a decision tree is used to solve the classification and regression problems.
10.	It uses the random subspace method and bagging during tree construction, which has built-in feature importance.	Whereas a decision is made based on the selected sample's feature, this is usually a feature that is used to make a decision, decision tree learning is a process to find the optimal value for each internal tree node.

Decision trees are simple but suffer from some serious problems- overfitting, error due to variance or error due to bias. Random Forest is the collection of decision trees with a single and aggregated result. Using multiple trees in the random forest reduces the chances of

overfitting. And they are complex to understand. A decision tree is easy to read and understand whereas random forest is more complicated to interpret. A single decision tree is not accurate in predicting the results but is fast to implement. More trees will give a more robust model and prevents overfitting. In the forest, we need to generate process and analyze each and every tree. Therefore this process is a slow process and can sometimes take hours or even days.

Mathematics based questions

14. Explain Bagging and Boosting in training random forest tree.

The Ensemble learning helps improve machine learning results by combining several models. This approach allows the production of better predictive performance compared to a single model. Basic idea is to learn a set of classifiers (experts) and to allow them to vote. **Bagging** and **Boosting** are two types of **Ensemble Learning**. These two decrease the variance of a single estimate as they combine several estimates from different models. So the result may be a model with higher stability. Let's understand these two terms in a glimpse.

1. **Bagging**: It is a homogeneous weak learners' model that learns from each other independently in parallel and combines them for determining the model average.
2. **Boosting**: It is also a homogeneous weak learners' model but works differently from Bagging. In this model, learners learn sequentially and adaptively to improve model predictions of a learning algorithm.

Bagging: Bootstrap **Agg**regating, also known as bagging, is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. It decreases the variance and helps to avoid overfitting. It is usually applied to decision tree methods. Bagging is a special case of the model averaging approach.

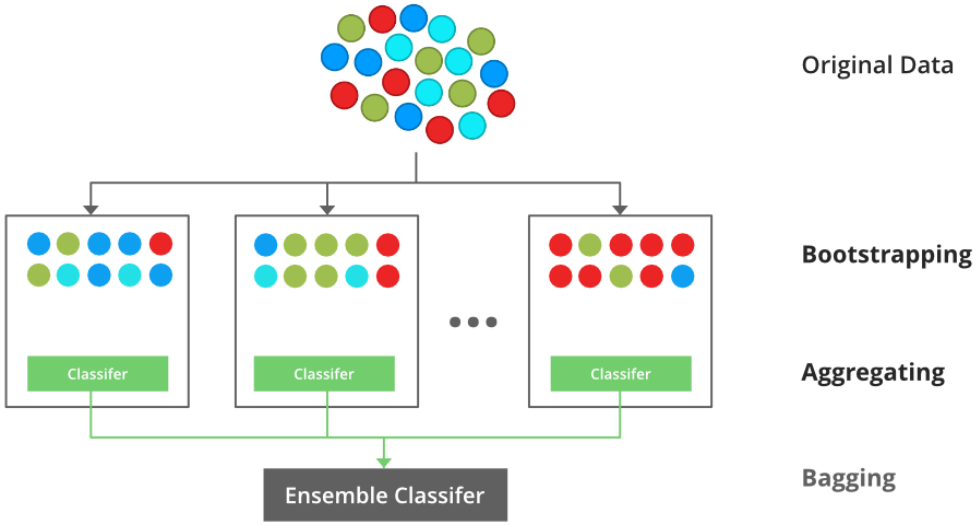
Description of the Technique

Suppose a set D of d tuples, at each iteration i, a training set D_i of d tuples is sampled with replacement from D (i.e., bootstrap). Then a classifier model Mi is learned for each training set D < i. Each classifier M_i returns its class prediction. The bagged classifier M* counts the votes and assigns the class with the most votes to X (unknown sample).

Implementation Steps of Bagging

- **Step 1:** Multiple subsets are created from the original data set with equal tuples, selecting observations with replacement.
- **Step 2:** A base model is created on each of these subsets.
- **Step 3:** Each model is learned in parallel from each training set and independent of each other.

- **Step 4:** The final predictions are determined by combining the predictions from all the models.



An illustration for the concept of bootstrap aggregating (Bagging)

Example of Bagging

The Random Forest model uses Bagging, where decision tree models with higher variance are present. It makes random feature selection to grow trees. Several random trees make a Random Forest.

Boosting

Boosting is an ensemble modelling technique that attempts to build a strong classifier from the number of weak classifiers. It is done by building a model by using weak models in series. Firstly, a model is built from the training data. Then the second model is built which tries to correct the errors present in the first model. This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models is added.

Boosting Algorithms

There are several boosting algorithms. The original ones, proposed by **Robert Schapire** and **Yoav Freund** were not adaptive and could not take full advantage of the weak learners. Schapire and Freund then developed AdaBoost, an adaptive boosting algorithm that won the prestigious Gödel Prize. AdaBoost was the first really successful boosting algorithm developed for the purpose of binary classification. AdaBoost is short for Adaptive Boosting and is a very popular boosting technique that combines multiple "weak classifiers" into a single "strong classifier".

Algorithm:

1. *Initialise the dataset and assign equal weight to each of the data point.*
2. *Provide this as input to the model and identify the wrongly classified data points.*
3. *Increase the weight of the wrongly classified data points.*

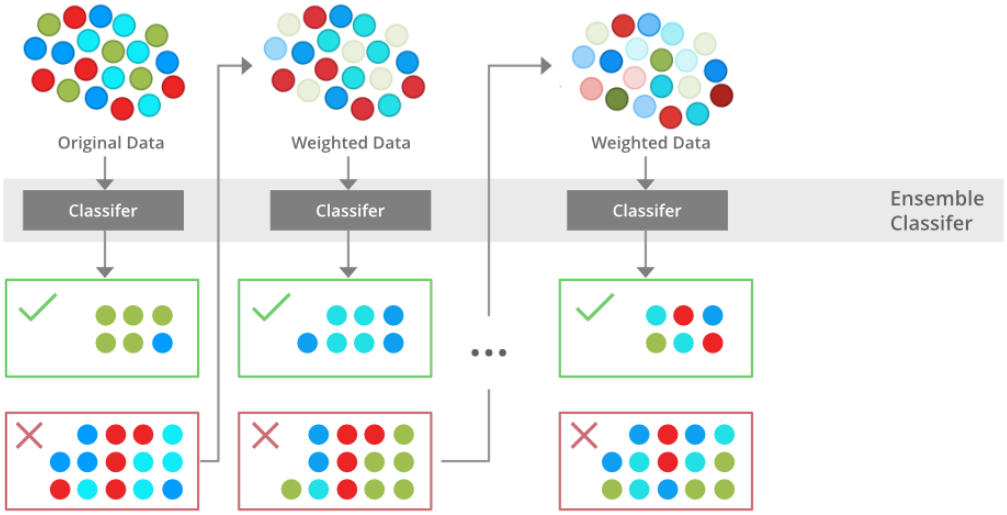
4. if (got required results)

Goto step 5

Else

Goto step 2

5. End



An illustration presenting the intuition behind the boosting algorithm, consisting of the parallel learners and weighted dataset

Similarities between Bagging and Boosting

Bagging and Boosting, both being the commonly used methods, have a universal similarity of being classified as ensemble methods. Here we will explain the similarities between them.

- Both are ensemble methods to get N learners from 1 learner.
- Both generate several training data sets by random sampling.
- Both make the final decision by averaging the N learners (or taking the majority of them i.e Majority Voting).
- Both are good at reducing variance and provide higher stability.

Differences between Bagging and Boosting

SN	Bagging	Boosting
1.	The simplest way of combining predictions that belongs to the same type.	A way of combining predictions that belong to the different types.
2.	Aim to decrease variance, not bias.	Aim to decrease bias, not variance.
3.	Each model receives equal weight.	Models are weighted according to their performance.
4.	Each model is built independently.	New models are influenced by the performance of previously built models.

5.	Different training data subsets are randomly drawn with replacement from the entire training dataset.	Every new subset contains the elements that were misclassified by previous models.
6.	Bagging tries to solve the over-fitting problem.	Boosting tries to reduce bias.
7.	If the classifier is unstable (high variance), then apply bagging.	If the classifier is stable and simple (high bias) the apply boosting.
8.	Example: The Random forest model uses Bagging.	Example: The AdaBoost uses Boosting techniques

Bagging (Bootstrap AGGregatING)

Given a training set $D = \{(\mathbf{x}_1, y_1), \ldots (\mathbf{x}_n, y_n)\}$,

- sample T sets of n elements from D (with replacement) $D_1, D_2, \ldots D_T \rightarrow T$ quasi replica training sets;
- train a machine on each D_i, $i = 1, ..., T$ and obtain a sequence of T outputs $f_1(\mathbf{x}), \ldots f_T(\mathbf{x})$.

Bagging (cont.)

The final aggregate classifier can be

- for regression

$$\bar{f}(\mathbf{x}) = \sum_{i=1}^{T} f_i(\mathbf{x}),$$

the average of f_i for $i = 1, ..., T$;

- for classification

$$\bar{f}(\mathbf{x}) = \text{sign}(\sum_{i=1}^{T} f_i(\mathbf{x}))$$

or the majority vote

$$\bar{f}(\mathbf{x}) = \text{sign}(\sum_{i=1}^{T} \text{sign}(f_i(\mathbf{x})))$$

The original Boosting (Schapire, 1990): For Classification Only

1. Train a first classifier f_1 on a training set drawn from a probability $p(\mathbf{x}, y)$. Let ϵ_1 be the obtained training performance;

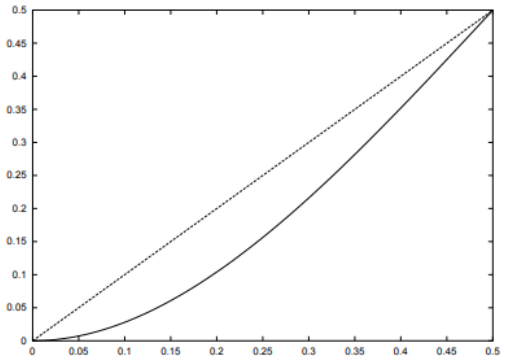
2. Train a second classifier f_2 on a training set drawn from a probability $p_2(\mathbf{x}, y)$ such that it has half its measure on the event that h_1 makes a mistake and half on the rest. Let ϵ_2 be the obtained performance;

3. Train a third classifier f_3 on disagreements of the first two – that is, drawn from a probability $p_3(\mathbf{x}, y)$ which has its support on the event that h_1 and h_2 disagree. Let ϵ_3 be the obtained performance.

Main result: If $\epsilon_i < p$ for all i, the boosted hypothesis

$$g = MajorityVote\ (f_1, f_2, f_3)$$

has training performance no worse than $\epsilon = 3p^2 - 2p^3$



Adaboost (Freund and Schapire, 1996)

The idea is of *adaptively* resampling the data

- Maintain a probability distribution over training set;

- Generate a sequence of classifiers in which the "next" classifier focuses on sample where the "previous" classifier failed;

- *Weigh* machines according to their performance.

Adaboost

Given: a class $\mathcal{F} = \{f : \mathcal{X} \mapsto \{-1, 1\}\}$ of weak learners and the data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, $y_i \in \{-1, 1\}$. Initialize the weights as $w_1(i) = 1/n$.
For $t = 1, \ldots T$:

1. Find a weak learner f_t based on weights $w_t(i)$;
2. Compute the *weighted* error $\epsilon_t = \sum_{i=1}^{n} w_t(i) I(y_i \neq f_t(x_i))$;
3. Compute the *importance* of f_t as $\alpha_t = 1/2 \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$;
4. Update the distribution $w_{t+1}(i) = \frac{w_t(i) e^{-\alpha_t y_i f_t(x_i)}}{Z_t}$, $Z_t = \sum_{i=1}^{n} w_t(i) e^{-\alpha_t y_i h_t(x_i)}$.

Adopt as final hypothesis

$$g(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t f_t(\mathbf{x})\right)$$

15. Which is the best, Bagging or Boosting?

- There's not an outright winner; it depends on the data, the simulation and the circumstances.
  Bagging and Boosting decrease the variance of your single estimate as they combine several estimates from different models. So the result may be a model with **higher stability**.
- If the problem is that the single model gets a very low performance, Bagging will rarely get a **better bias**. However, Boosting could generate a combined model with lower errors as it optimises the advantages and reduces pitfalls of the single model.
- By contrast, if the difficulty of the single model is **over-fitting**, then Bagging is the best option. Boosting for its part doesn't help to avoid over-fitting; in fact, this technique is faced with this problem itself. Thus, Bagging is effective more often than Boosting.

16. What are the main advantages of using a random forest versus a single decision tree?

In an ideal world, we'd like to reduce both bias-related and variance-related errors. This issue is well-addressed by random forests. A random forest is nothing more than a series of decision trees with their findings combined into a single final result. They are so powerful because of their capability to reduce overfitting without massively increasing error due to

bias. Random forests, on the other hand, are a powerful modelling tool that is far more resilient than a single decision tree. They combine numerous decision trees to reduce overfitting and bias-related inaccuracy, and hence produce usable results.

Topic: Naive Bayes

Theory	Mathematics	Numerical
✓	✓	✓

Theory questions

17. Why use Naive Bayes 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**.

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.

18. What are the Pros and Cons of using Naive Bayes?

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.

- The requirement of predictors to be independent. In most of the real life cases, the predictors are dependent, this hinders the performance of the classifier.

19. How does the Bayes algorithm differ from decision trees?

- Decision tree is a discriminative model, whereas Naive bayes is a generative model.
- Decision trees are more flexible and easy. Decision tree pruning may neglect some key values in training data, which can lead the accuracy for a toss.
- A major advantage to Naive Bayes classifiers is that they are not prone to overfitting, thanks to the fact that they "ignore" irrelevant features. **They are, however, prone to poisoning, a phenomenon that occurs when we are trying to predict a class but features uncommon to appear, causing a misclassification.**
- Naive Bayes classifiers are easily implemented and highly scalable, with a linear computational complexity with respect to the number of data entries.
- Naive Bayes is strongly associated with text-based classification. Example applications include but are not limited to spam filtering and text categorization. This is because the presence of certain words is strongly linked to their respective categories, and thus the mutual independence assumption is stronger.
- Unfortunately, several data sets require that some features are hand-picked before the classifier can work as intended.
- Decision Trees are very flexible, easy to understand, and easy to debug. They will work with classification problems and regression problems. So if you are trying to predict a categorical value like (red, green, up, down) or if you are trying to predict a continuous value like 2.9, 3.4 etc Decision Trees will handle both problems. Probably one of the coolest things about Decision Trees is they only need a table of data and they will build a classifier directly from that data without needing any up front design work to take place. To some degree properties that don't matter won't be chosen as splits and will get eventually pruned so it's very tolerant of nonsense. To start it's set it and forget it.
- However, the downside. Simple decision trees tend to over fit the training data more so that other techniques which mean you generally have to do tree pruning and tune the pruning procedures. You didn't have any upfront design cost, but you'll pay that back on tuning the trees performance. Also simple decision trees divide the data into squares so building clusters around things means it has to split a lot to encompass clusters of data. Splitting a lot leads to complex trees and raises probability you are overfitting. Tall trees get pruned back so while you can build a cluster around some feature in the data it might not survive the pruning process. There are other techniques like **surrogate splits** which let you split along several variables at once creating splits in the space that aren't either horizontal or perpendicular (0 < slope < infinity). Cool, but your tree starts to

become harder to understand, and its complex to implement these algorithms. Other techniques like boosting and random forest decision trees can perform quite well, and some feel these techniques are essential to get the best performance out of decision trees. Again this adds more things to understand and use to tune the tree and hence more things to implement. In the end the more we add to the algorithm the taller the barrier to using it.

- Naive Bayes requires you build a classification by hand. There's not way to just toss a bunch of tabular data at it and have it pick the best features it will use to classify. Picking which features matter is up to you. Decisions trees will pick the best features for you from tabular data. If there were a way for Naive Bayes to pick features you'd be getting close to using the same techniques that make decision trees work like that. Give this fact that means you may need to combine Naive Bayes with other statistical techniques to help guide you towards what features best classify and that could be using decision trees. Naive bayes will answer as a continuous classifier. There are techniques to adapt it to categorical prediction however they will answer in terms of probabilities like (A 90%, B 5%, C 2.5% D 2.5%) Bayes can perform quite well, and it doesn't over fit nearly as much so there is no need to prune or process the network. That makes them simpler algorithms to implement. However, they are harder to debug and understand because it's all probabilities getting multiplied 1000's of times so you have to be careful to test it's doing what you expect. Naive bayes does quite well when the training data doesn't contain all possibilities so it can be very good with low amounts of data. Decision trees work better with lots of data compared to Naive Bayes.
- Naive Bayes is used a lot in robotics and computer vision, and does quite well with those tasks. Decision trees perform very poorly in those situations. Teaching a decision tree to recognize poker hands by looking a millions of poker hands does very poorly because royal flushes and quads occurs so little it often gets pruned out. If it's pruned out of the resulting tree it will misclassify those important hands (recall tall trees discussion from above). Now just think if you are trying to diagnose cancer using this. Cancer doesn't occur in the population in large amounts, and it will get pruned out more likely. Good news is this can be handled by using weights so we weight a winning hand or having cancer as higher than a losing hand or not having cancer and that boosts it up the tree so it won't get pruned out. Again this is the part of tuning the resulting tree to the situation that I discussed earlier.
- Decision trees are neat because they tell you what inputs are the best predicators of the outputs so often decision trees can guide you to find if there is a statistical relationship between a given input to the output and how strong that relationship is. Often the

resulting decision tree is less important than relationships it describes. So decision trees can be used a research tool as you learn about your data so you can build other classifiers.

Mathematics based questions

20. How Naive Bayes Algorithms works?

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:

SN	Outlook	Play	SN	Outlook	Play	SN	Outlook	Play	SN	Outlook	Play
0	Rainy	Yes	**4**	Sunny	No	**8**	Rainy	No	**12**	Overcast	Yes
1	Sunny	Yes	**5**	Rainy	Yes	**9**	Sunny	No	**13**	Overcast	Yes
2	Overcast	Yes	**6**	Sunny	Yes	**10**	Sunny	Yes			
3	Overcast	Yes	**7**	Overcast	Yes	**11**	Rainy	No			

Frequency table for the Weather Conditions:

Weather	Yes	No
Overcast	5	0
Rainy	2	2
Sunny	3	2
Total	**10**	**4**

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.

21. Explain 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. Using Bayes theorem, we can find the probability of **A** happening, given that **B** has occurred. Here, **B** is the evidence and **A** is the hypothesis. The assumption made here is that the predictors/features are independent. That is presence of one particular feature does not affect the other. Hence it is called naive.

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

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.

Problems/Numerical

22. Problems on application of Bayes theorem for classification

Problem 1:

Consider a car theft example. The attributes are Colour, Type, Origin, and the subject, stolen can be either yes or no. Use Naive Bayes Classifier to classify a "Red Domestic SUV".

Dataset is as below.

Example No.	Color	Type	Origin	Stolen?
1	Red	Sports	Domestic	Yes
2	Red	Sports	Domestic	No
3	Red	Sports	Domestic	Yes
4	Yellow	Sports	Domestic	No
5	Yellow	Sports	Imported	Yes
6	Yellow	SUV	Imported	No
7	Yellow	SUV	Imported	Yes
8	Yellow	SUV	Domestic	No
9	Red	SUV	Imported	No
10	Red	Sports	Imported	Yes

Solution:

The Bayes Naive classifier selects the most likely classification V_{nb} given the attribute values $a_1, a_2, \ldots a_n$. This results in:

$$V_{nb} = \text{argmax}_{v_j \in V} P(v_j) \prod P(a_i|v_j)$$

We generally estimate $P(a_i|v_j)$ using m-estimates:

$$P(a_i|v_j) = \frac{n_c + mp}{n + m}$$

where:

$n =$ the number of training examples for which $v = v_j$
$n_c =$ number of examples for which $v = v_j$ and $a = a_i$
$p =$ a priori estimate for $P(a_i|v_j)$
$m =$ the equivalent sample size

Note there is no example of a Red Domestic SUV in our data set. We need to calculate the probabilities P(Red|Yes), P(SUV|Yes), P(Domestic|Yes) , P(Red|No) , P(SUV|No), and P(Domestic|No) and multiply them by P(Yes) and P(No) respectively .

```
Yes:                    No:
    Red:                    Red:
        n = 5                   n = 5
        n_c= 3                  n_c = 2
        p = .5                  p = .5
        m = 3                   m = 3
    SUV:                    SUV:
        n = 5                   n = 5
        n_c = 1                 n_c = 3
        p = .5                  p = .5
        m = 3                   m = 3
    Domestic:               Domestic:
        n = 5                   n = 5
        n_c = 2                 n_c = 3
        p = .5                  p = .5
        m = 3                   m =3
```

Looking at $P(Red|Yes)$, we have 5 cases where v_j = Yes , and in 3 of those cases a_i = Red. So for $P(Red|Yes)$, $n = 5$ and $n_c = 3$. Note that all attribute are binary (two possible values). We are assuming no other information so, $p = 1$ / (number-of-attribute-values) = 0.5 for all of our attributes. Our m value is arbitrary, (We will use $m = 3$) but consistent for all attributes. Now we simply apply eqauation (3) using the precomputed values of n , n_c, p, and m.

$$P(Red|Yes) = \frac{3 + 3 * .5}{5 + 3} = .56 \qquad P(Red|No) = \frac{2 + 3 * .5}{5 + 3} = .43$$
$$P(SUV|Yes) = \frac{1 + 3 * .5}{5 + 3} = .31 \qquad P(SUV|No) = \frac{3 + 3 * .5}{5 + 3} = .56$$
$$P(Domestic|Yes) = \frac{2 + 3 * .5}{5 + 3} = .43 \qquad P(Domestic|No) = \frac{3 + 3 * .5}{5 + 3} = .56$$

We have $P(Yes) = .5$ and $P(No) = .5$, so we can apply equation (2). For $v = Yes$, we have

```
P(Yes) * P(Red | Yes) * P(SUV | Yes) * P(Domestic|Yes)

    =  .5 * .56 * .31 * .43   = .037
```

and for $v = No$, we have

```
P(No) * P(Red | No) * P(SUV | No) * P (Domestic | No)

    = .5 * .43 * .56 * .56 = .069
```

Since $0.069 > 0.037$, our example gets classified as 'NO'

Problem 2:

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

A: attributes

M: mammals

N: non-mammals

$$P(A \mid M) = \frac{6}{7} \times \frac{6}{7} \times \frac{2}{7} \times \frac{2}{7} = 0.06$$

$$P(A \mid N) = \frac{1}{13} \times \frac{10}{13} \times \frac{3}{13} \times \frac{4}{13} = 0.0042$$

$$P(A \mid M)P(M) = 0.06 \times \frac{7}{20} = 0.021$$

$$P(A \mid N)P(N) = 0.004 \times \frac{13}{20} = 0.0027$$

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

P(A|M)P(M) > P(A|N)P(N)

=> Mammals

Problem 3:

The weather data, with counts and probabilities

outlook			temperature			humidity			windy		play	
	yes	no		yes	no		yes	no		yes	no	yes	no
sunny	2	3	hot	2	2	high	3	4	false	6	2	9	5
overcast	4	0	mild	4	2	normal	6	1	true	3	3		
rainy	3	2	cool	3	1								
sunny	2/9	3/5	hot	2/9	2/5	high	3/9	4/5	false	6/9	2/5	9/14	5/14
overcast	4/9	0/5	mild	4/9	2/5	normal	6/9	1/5	true	3/9	3/5		
rainy	3/9	2/5	cool	3/9	1/5								

A new day

outlook	temperature	humidity	windy	play
sunny	cool	high	true	?

- Likelihood of yes

$$= \frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{9}{14} = 0.0053$$

- Likelihood of no

$$= \frac{3}{5} \times \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5} \times \frac{5}{14} = 0.0206$$

- Therefore, the prediction is No

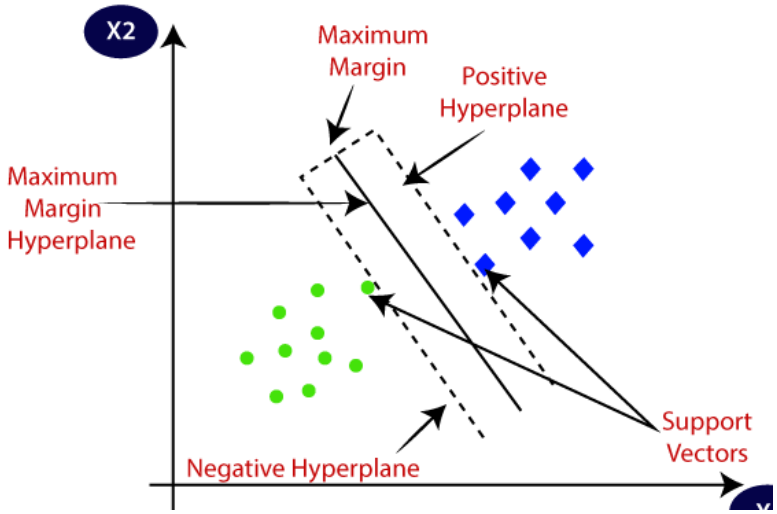
Topic: Support Vector Machine

Theory	Mathematics	Numerical
✓	✓	✓

Theory questions

23. What is Support Vector Machine?

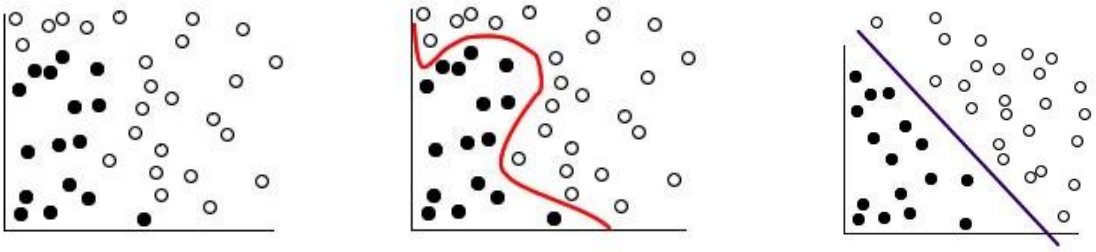
Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning. The goal of the SVM algorithm is to create the best line or decision 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. 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:



24. How does the SVM work?

SVM works by mapping data to a high-dimensional feature space so that data points can be categorized, even when the data are not otherwise linearly separable.

Original dataset | Data with separator added | Transformed data



A separator between the categories is found, and then the data are transformed in such a way that the separator could be drawn as a hyperplane. Following this, characteristics of new

data can be used to predict the group to which a new record should belong. For example, consider the following figure, in which the data points fall into two different categories. The two categories can be separated with a curve, as shown in the figure. After the transformation, the boundary between the two categories can be defined by a hyperplane, as shown in the following figure.

The mathematical function used for the transformation is known as the kernel function. Following are the popular functions.

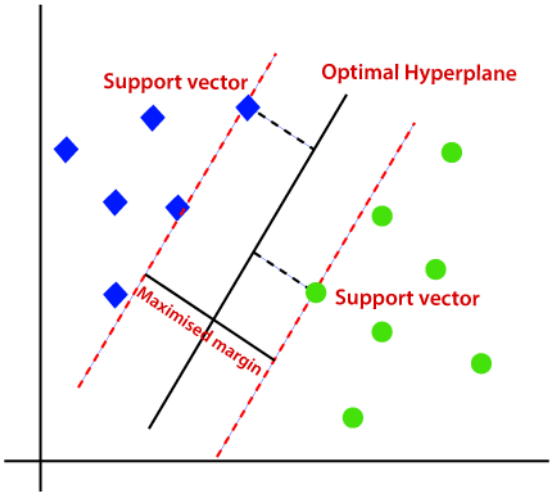
- Linear
- Polynomial
- Radial basis function (RBF)
- Sigmoid

A linear kernel function is recommended when linear separation of the data is straightforward. In other cases, one of the other functions should be used. You will need to experiment with the different functions to obtain the best model in each case, as they each use different algorithms and parameters.

25. Explain Hyperplanes and Support Vectors.

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. 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.

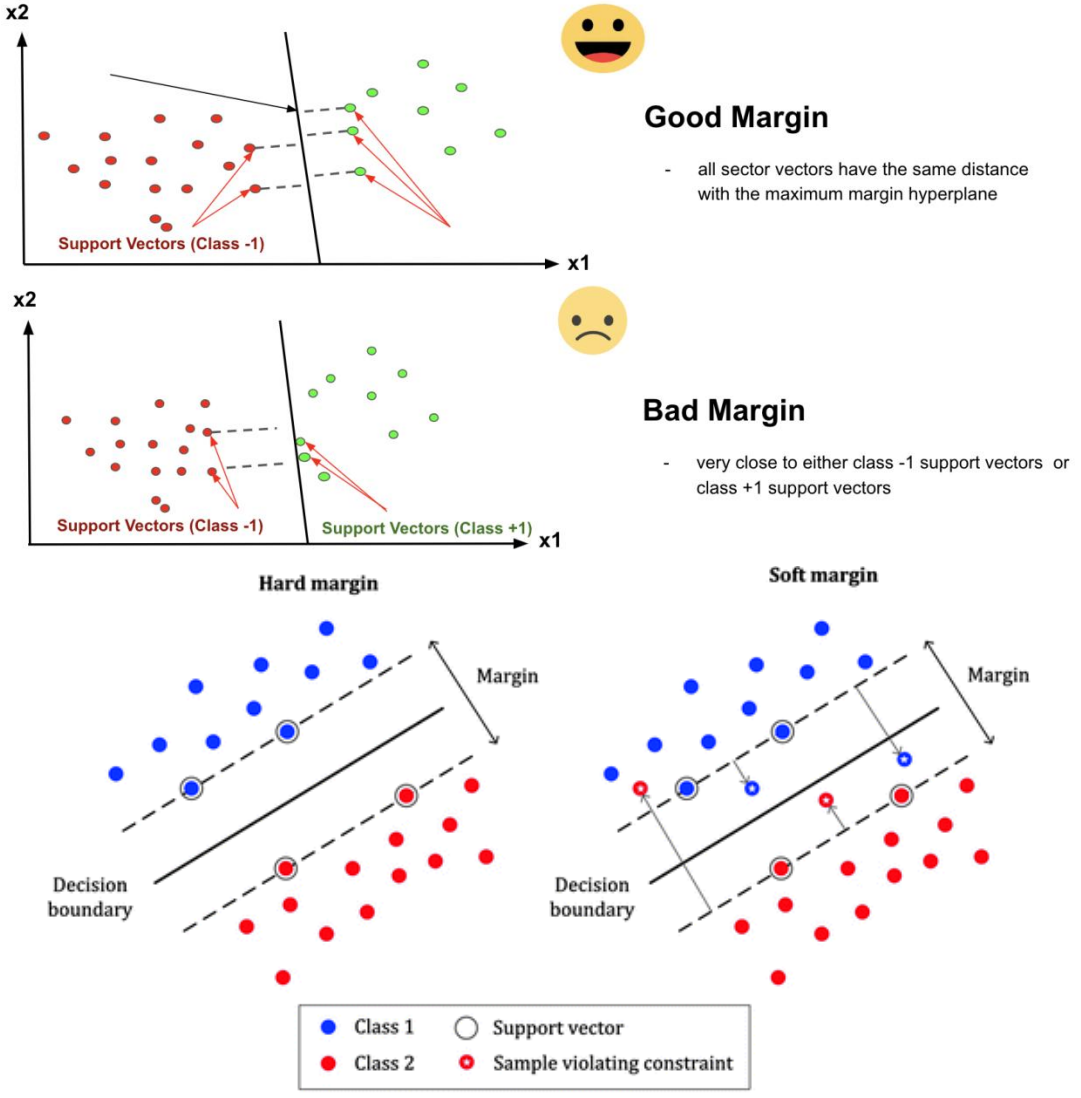
Support Vectors: The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.



26. Explain Hard and soft margins with the help of sketch

The distance of the vectors from the hyperplane is called the **margin** which is a separation of a line to the closest class points. We would like to choose a hyperplane that maximizes the margin between classes. The graph below shows what good margins and bad margins are. Again Margin can be sub-divided into,

- **Soft Margin** – As most of the real-world data are not fully linearly separable, we will allow some margin violation to occur which is called soft margin classification. It is better to have a large margin, even though some constraints are violated. Margin violation means choosing a hyperplane, which can allow some data points to stay on either the incorrect side of the hyperplane and between the margin and correct side of the hyperplane.
- **2. Hard Margin** – If the training data is linearly separable, we can select two parallel hyperplanes that separate the two classes of data, so that the distance between them is as large as possible.



27. Explain Support Vector Machine terminology.

Support Vector Machines are part of the supervised learning model with an associated learning algorithm. It is the most powerful and flexible algorithm used for classification, regression, and detection of outliers. It is used in case of high dimension spaces; where each data item is plotted as a point in n-dimension space such that each feature value corresponds to the value of specific coordinate. The classification is made on the basis of a hyperplane/line as wide as possible, which distinguishes between two categories more clearly. Basically, support vectors are the observational points of each individual, whereas the support vector machine is the boundary that differentiates one class from another class.

Some significant terminology of SVM is given below:

- **Support Vectors:** These are the data point or the feature vectors lying nearby to the hyperplane. These help in defining the separating line.
- **Hyperplane:** It is a subspace whose dimension is one less than that of a decision plane. It is used to separate different objects into their distinct categories. The best hyperplane is the one with the maximum separation distance between the two classes.
- **Margins:** It is defined as the distance (perpendicular) from the data point to the decision boundary. There are two types of margins: good margins and margins. **Good margins** are the one with huge margins and the **bad margins** in which the margin is minor.

The main goal of SVM is to find the maximum marginal hyperplane, so as to segregate the dataset into distinct classes. It undergoes the following steps:

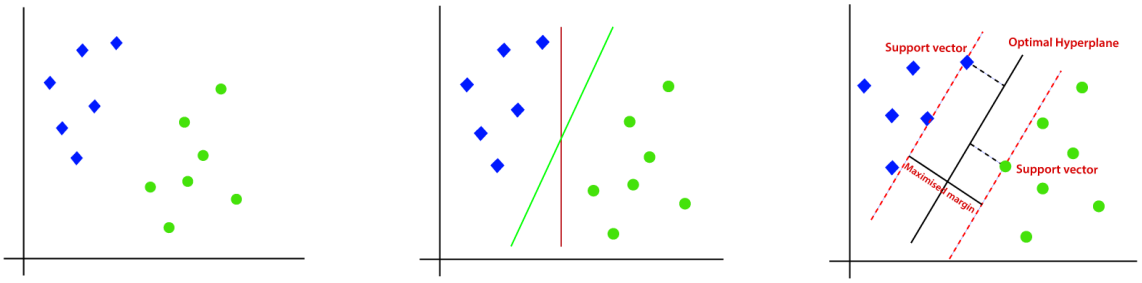
- Firstly the SVM will produce the hyperplanes repeatedly, which will separate out the class in the best suitable way.
- Then we will look for the best option that will help in correct segregation.

28. Explain Linear SVM, non-linear SVM.

Linear SVM:

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. Consider the below image: 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. Consider the below image. 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 maximum margin is called the **optimal hyperplane**.



Non-Linear SVM:

If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:

So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third dimension z. It can be calculated as:

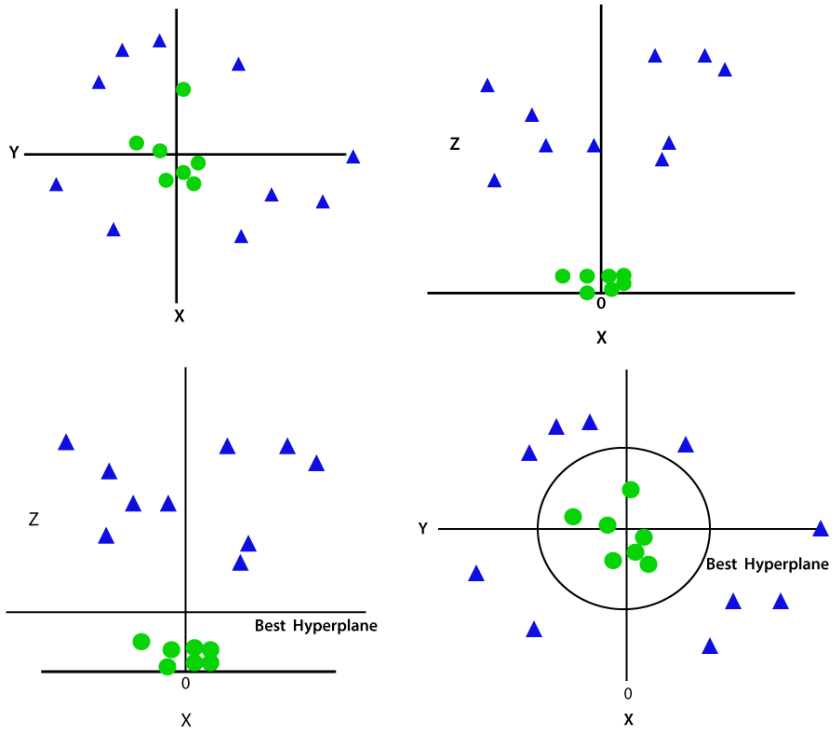
$z=x^2 +y^2$

By adding the third dimension, the sample space will become as below image:

So now, SVM will divide the datasets into classes in the following way. Consider the below image:

Since we are in 3-d Space, hence it is looking like a plane parallel to the x-axis. If we convert it in 2d space with z=1, then it will become as:

Hence we get a circumference of radius 1 in case of non-linear data.



29. What are advantages and limitations of the Support Vector Machine

Advantages

- SVM's are very good when we have no idea on the data.
- Works well with even unstructured and semi structured data like text, Images and trees.
- The kernel trick is real strength of SVM. With an appropriate kernel function, we can solve any complex problem.
- Unlike in neural networks, SVM is not solved for local optima.
- It scales relatively well to high dimensional data. SVM is more effective in high dimensional spaces.
- SVM models have generalization in practice; the risk of over-fitting is less in SVM.
- SVM works relatively well when there is a clear margin of separation between classes.
- SVM is effective in cases where the number of dimensions is greater than the number of samples.
- SVM is relatively memory efficient

Disadvantages

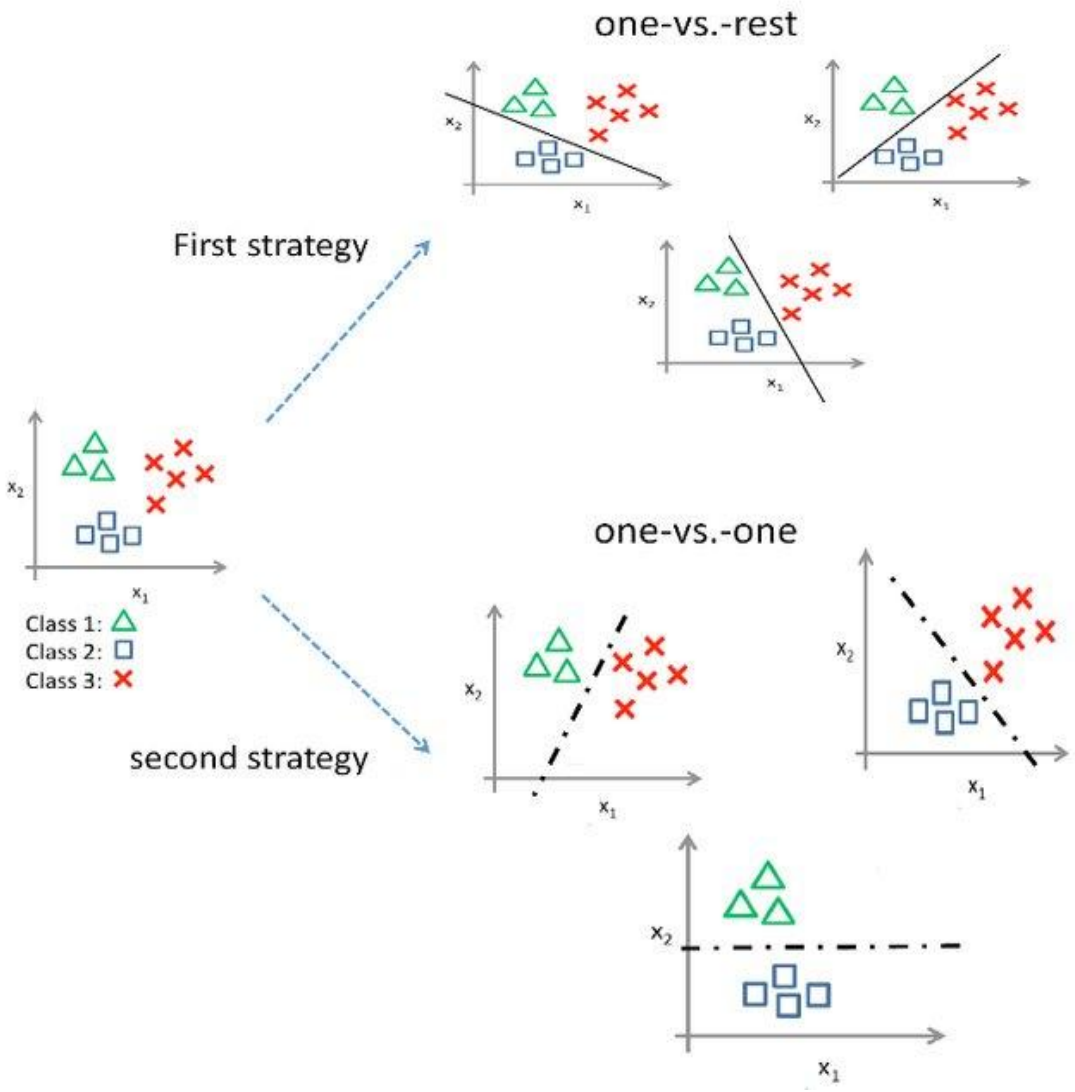
- Choosing a "good" kernel function is not easy.
- SVM algorithm is not suitable for large data sets.
- Long training time for large datasets.
- Difficult to understand and interpret the final model, variable weights and individual impact.
- Since the final model is not so easy to see, we cannot do small calibrations to the model hence it's tough to incorporate our business logic.
- The SVM hyper parameters are Cost -C and gamma. It is not that easy to fine-tune these hyper-parameters. It is hard to visualize their impact
- SVM does not perform very well when the data set has more noise i.e. target classes are overlapping.
- In cases where the number of features for each data point exceeds the number of training data samples, the SVM will underperform.
- As the support vector classifier works by putting data points, above and below the classifying hyperplane there is no probabilistic explanation for the classification.

30. Explain multi class classification methods

Two different examples of this approach are the One-vs-Rest and One-vs-One strategies.

- Binary classification models like logistic regression and SVM do not support multi-class classification natively and require meta-strategies.

- The One-vs-Rest strategy splits a multi-class classification into one binary classification problem per class.
- The One-vs-One strategy splits a multi-class classification into one binary classification problem per each pair of classes.



31. Explain hyper parameters of SVM.

Hyper parameters of SVM are considered as Kernel, Regularization, Gamma and Margin.

Kernel: The learning of the hyperplane in linear SVM is done by transforming the problem using some linear algebra. This is where the kernel plays role.

For **linear kernel** the equation for prediction for a new input using the dot product between the input (x) and each support vector (xi) is calculated as follows:

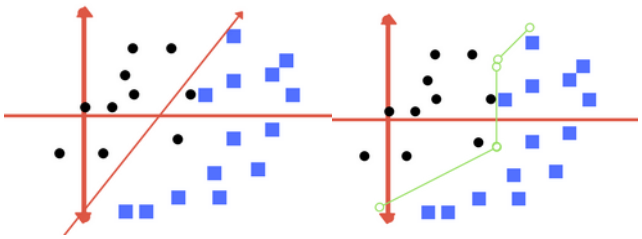
f(x) = B(0) + sum(ai * (x,xi))

This is an equation that involves calculating the inner products of a new input vector (x) with all support vectors in training data. The coefficients B0 and ai (for each input) must be estimated from the training data by the learning algorithm.

The **polynomial kernel** can be written as *K(x,xi) = 1 + sum(x * xi)^d* and **exponential** as *K(x,xi) = exp(-gamma * sum((x — xi²)).*

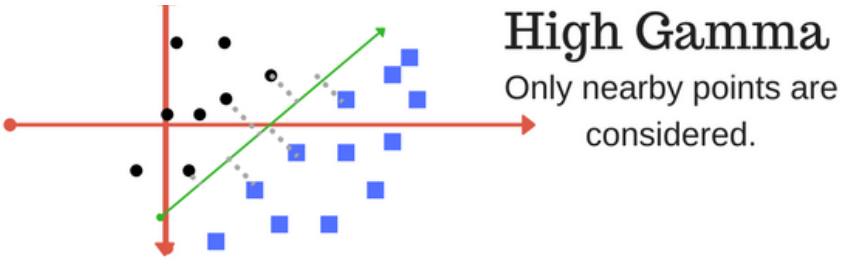
Polynomial and exponential kernels calculates separation line in higher dimension. This is called **kernel trick.**

Regularization: The Regularization parameter (often termed as C parameter in python's sklearn library) tells the SVM optimization how much you want to avoid misclassifying each training example. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points. The images below are example of two different regularization parameters. Left one has some misclassification due to lower regularization value. Higher value leads to results like right one.

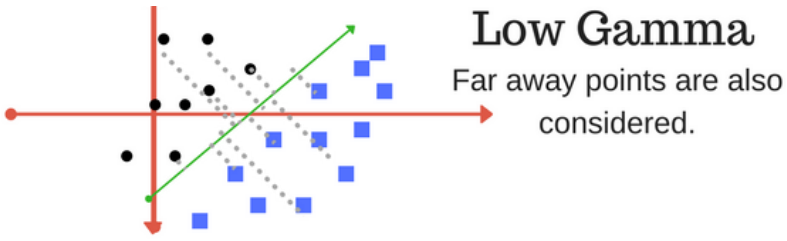


Left: low regularization value, right: high regularization value

Gamma: The gamma parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'. In other words, with low gamma, points far away from plausible separation line are considered in calculation for the separation line. Whereas high gamma means the points close to plausible line are considered in calculation.



High Gamma

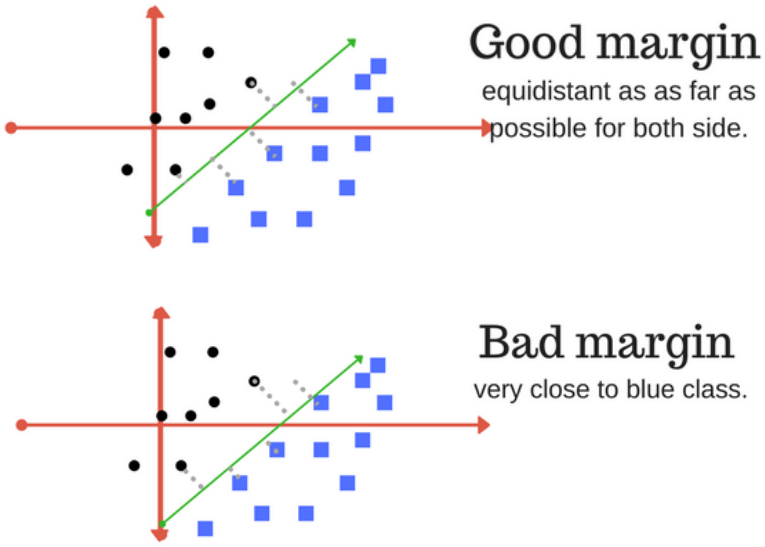


Low Gamma

Margin: And finally last but very important characteristic of SVM classifier. SVM to core tries to achieve a good margin.

A margin is a separation of line to the closest class points.

A ***good margin*** is one where this separation is larger for both the classes. Images below gives to visual example of good and bad margin. A good margin allows the points to be in their respective classes without crossing to other class.



Hyper parameters in detail

Sr.No	Parameter & Description
1	***C*** *– float, optional, default = 1.0* It is the penalty parameter of the error term.
2	***kernel*** *– string, optional, default = 'rbf'* This parameter specifies the type of kernel to be used in the algorithm. we can choose any one among, **'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'**. The default value of kernel would be **'rbf'**.
3	***degree*** *– int, optional, default = 3* It represents the degree of the 'poly' kernel function and will be ignored by all other kernels.
4	***gamma*** *– {'scale', 'auto'} or float,* It is the kernel coefficient for kernels 'rbf', 'poly' and 'sigmoid'.
5	***optinal default*** *– = 'scale'* If you choose default i.e. gamma = 'scale' then the value of gamma to be used by SVC is $1/(n\_features * X.var())$. On the other hand, if gamma= 'auto', it uses $1/n\_features$.
6	***coef0*** *– float, optional, Default=0.0* An independent term in kernel function which is only significant in 'poly' and 'sigmoid'.

7	***tol*** *– float, optional, default = 1.e-3* This parameter represents the stopping criterion for iterations.
8	***shrinking*** *– Boolean, optional, default = True* This parameter represents that whether we want to use shrinking heuristic or not.
9	***verbose*** *– Boolean, default: false* It enables or disable verbose output. Its default value is false.
10	***probability*** *– boolean, optional, default = true* This parameter enables or disables probability estimates. The default value is false, but it must be enabled before we call fit.
11	***max_iter*** *– int, optional, default = -1* As name suggest, it represents the maximum number of iterations within the solver. Value -1 means there is no limit on the number of iterations.
12	***cache_size*** *– float, optional* This parameter will specify the size of the kernel cache. The value will be in MB(MegaBytes).
13	***random_state*** *– int, RandomState instance or None, optional, default = none* This parameter represents the seed of the pseudo random number generated which is used while shuffling the data. Followings are the options – • **int** – In this case, *random_state* is the seed used by random number generator. • **RandomState instance** – In this case, random_state is the random number generator. • **None** – In this case, the random number generator is the RandonState instance used by np.random.
14	***class_weight*** *– {dict, 'balanced'}, optional* This parameter will set the parameter C of class j to $class\_weigh[j]*C$ for SVC. If we use the default option, it means all the classes are supposed to have weight one. On the other hand, if you choose ***class_weight:balanced***, it will use the values of y to automatically adjust weights.
15	***decision_function_shape*** *– ovo', 'ovr', default = 'ovr'* This parameter will decide whether the algorithm will return **'ovr'** (one-vs-rest) decision function of shape as all other classifiers, or the original **ovo**(one-vs-one) decision function of libsvm.
16	***break_ties*** *– boolean, optional, default = false* **True** – The predict will break ties according to the confidence values of decision_function **False** – The predict will return the first class among the tied classes.

Thus hyperparameter tuning is choosing a set of optimal hyperparameters for a learning algorithm. A hyperparameter is a model argument whose value is set before the learning process begins. The key to machine learning algorithms is hyperparameter tuning.

Mathematics based questions

32. Explain following Kernel functions: linear, polynomial, rbf, sigmoid

In this case, SVM introduces the kernel function
$K(x_n, x_i)$, which transforms the original data space into a new space with a higher dimension; this process includes the transformation function with dot product $\phi(x)$ (Equation (6)). The aim is the data, which already transformed into a higher dimension, can be separated easily. Thus the hyperplane function can be written in Equation (7).

$$K(x_n, x_i) = \phi(x_n)\phi(x_i) \tag{6}$$

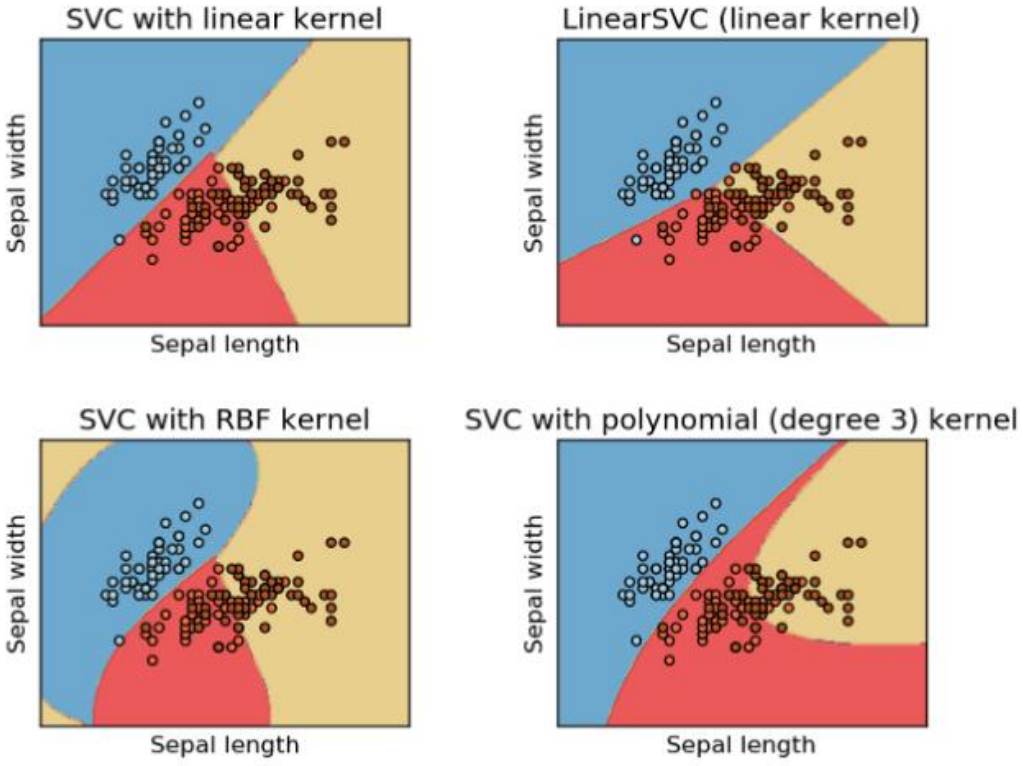
$$f(x_i) = \sum_{n=1}^{N} \alpha_n y_n K(x_n, x_i) + b \tag{7}$$

where, x_n is support vector data, α_n is lagrange multiplier and y_n is the label of membership class $(+1, -1)$ with $n = 1, 2, 3, \ldots, N$. In this study, we investigate the comparison of using the four kernel functions at the SVM algorithm, i.e., linear, radial basis function (RBF), sigmoid and polynomial, which are listed in Table 1.

Table 1. Four common kernels

No.	Kernel Function	Formula	Optimization Parameter
1	Linear	$K(x_n, x_i) = (x_n, x_i)$	C and γ
2	RBF	$K(x_n, x_i) = \exp\left(-\gamma \lVert x_n - x_i \rVert^2 + C\right)$	C and γ
3	Sigmoid	$K(x_n, x_i) = \tanh(\gamma(x_n, x_i) + r)$	C, γ, and r
4	Polynomial	$K(x_n, x_i) = (\gamma(x_n, x_i) + r)^d$	C, γ, r, and d

Explanation, C: cost; γ: gamma; r: coefficient; d: degree.



Problems/Numerical

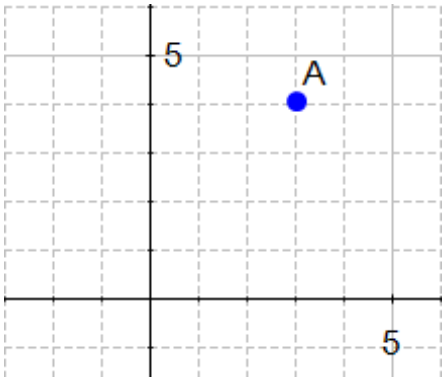
33. Problems based on calculating hyperplane and margin

In Support Vector Machine, there is the word **vector.** That means it is important to understand vector well and how to use them.

- What is a vector?
  - its norm
  - its direction
- How to add and subtract vectors?
- What is the dot product?
- How to project a vector onto another?
- What is the equation of the hyperplane?
- How to compute the margin?

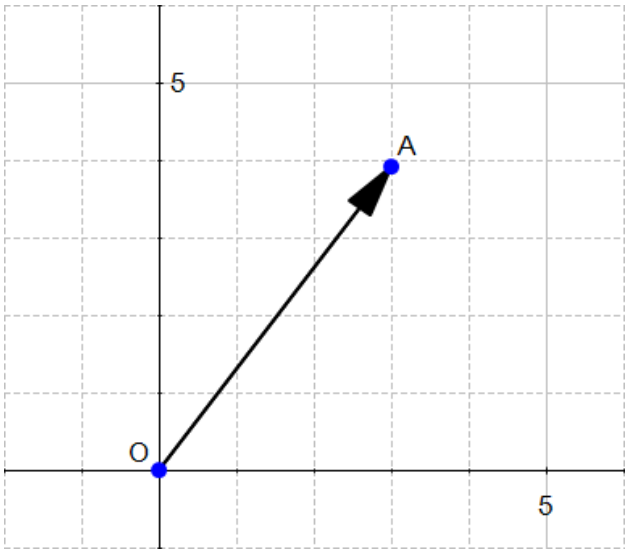
What is a vector?

If we define a point $A(3,4)$ in $\mathbb{R}^2$ we can plot it like this.



Definition: Any point $x=(x1, x2), x\neq 0$, in $\mathbb{R}^2$ specifies a vector in the plane, namely the vector starting at the origin and ending at x.

This definition means that there exists a vector between the origin and A.



If we say that the point at the origin is the point $O(0,0)$ then the vector above is the vector $\overrightarrow{OA}$. We could also give it an arbitrary name such as **u**.

Note:

You can notice that we write vector either with an arrow on top of them, or in bold, in the rest of this text I will use the arrow when there is two letters like $\overrightarrow{OA}$ and the bold notation otherwise.

Ok so now we know that there is a vector, but we still don't know what **IS** a vector.

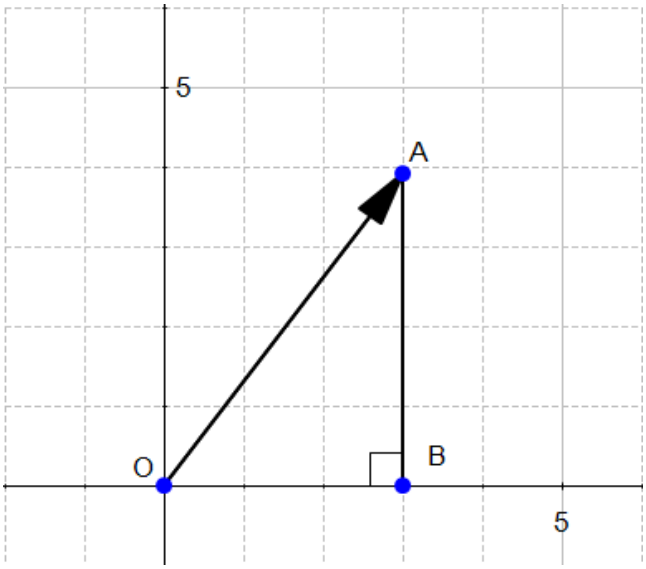
Definition: A vector is an object that has both a magnitude and a direction.

We will now look at these two concepts.

1) The magnitude

The magnitude or length of a vector x is written $\|x\|$ and is called its norm.

For our vector $\overrightarrow{OA}$, $\|OA\|$ is the length of the segment OA



From Figure, we can easily calculate the distance OA using Pythagoras' theorem:

$OA^2 = OB^2 + AB^2$

$OA^2 = 3^2 + 4^2$

$OA^2 = 25$

$OA = 5$

$\|OA\| = 5$

2) The direction

The direction is the second component of a vector.

Definition: The **direction** of a vector $\mathbf{u}(u_1, u_2)$ is the vector $\frac{u_1}{\|u\|}, \frac{u_2}{\|u\|}$

Where does the coordinates of **w** come from?

Understanding the definition

To find the direction of a vector, we need to use its angles.

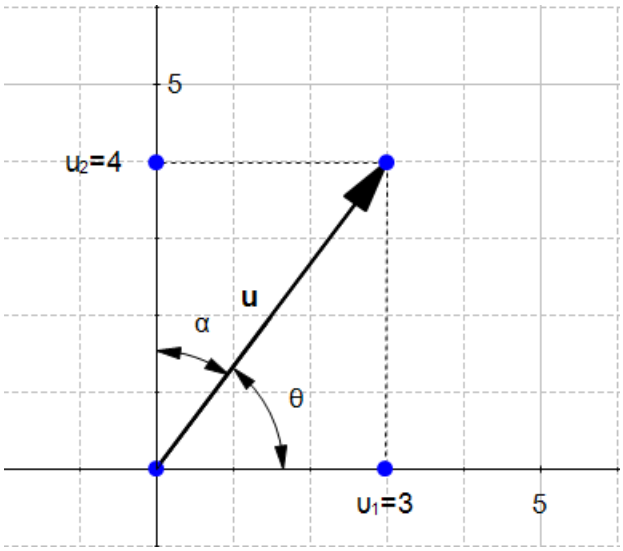


Figure 4 displays the vector **u**($u1$,$u2$) with $u1$=3 and $u2$=4

We could say that:

*Naive definition 1: The direction of the vector **u** is defined by the angle θ with respect to the horizontal axis, and with the angle α with respect to the vertical axis.* This is tedious. Instead of that we will use the cosine of the angles.

In a right triangle, the cosine of an angle β is defined by :

$cos(\beta) = adjacent/hypotenuse$

In Figure 4 we can see that we can form two right triangles, and in both case the adjacent side will be on one of the axis. Which means that the definition of the cosine implicitly contains the axis related to an angle. We can rephrase our naïve definition to :

*Naive definition 2: The direction of the vector **u** is defined by the cosine of the angle θ and the cosine of the angle α.*

Now if we look at their values:

$cos(\theta) = \frac{u_1}{\|u\|}$

$cos(\alpha) = \frac{u_2}{\|u\|}$

Hence the original definition of the vector ***w***. That's why its coordinates are also called *direction cosine*.

Computing the direction vector

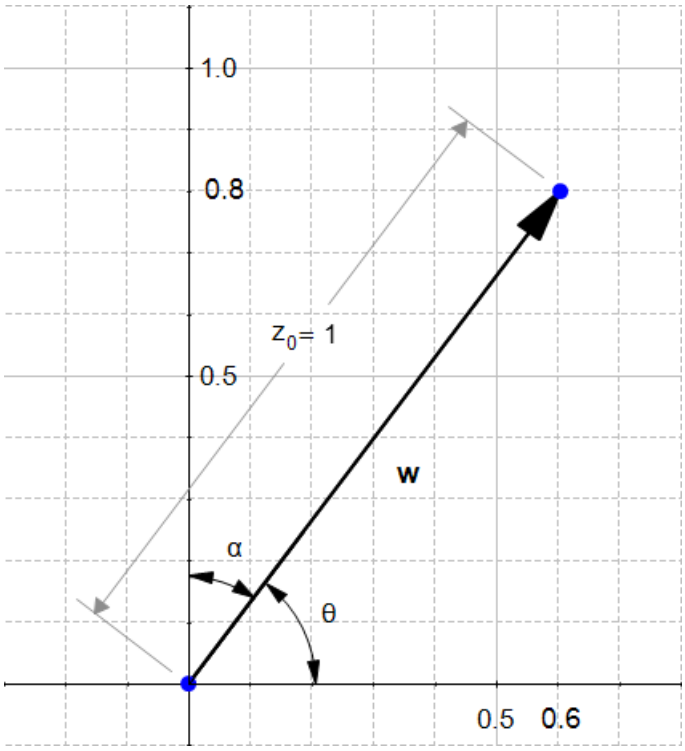
We will now compute the direction of the vector **u** from Figure 4.

$cos(\theta) = \frac{u_1}{\|u\|} = 3/5 = 0.6$

$cos(\alpha) = \frac{u_2}{\|u\|} = 4/5 = 0.8$

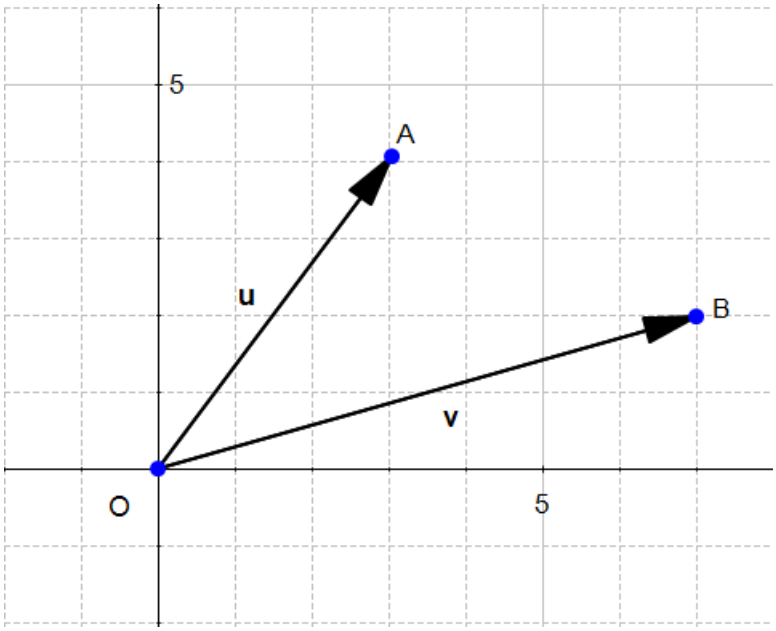
The direction of **u**(3,4) is the vector **w**(0.6,0.8)

If we draw this vector we get Figure 5:



We can see that '**w'** as indeed the same look as **u** except it is smaller. Something interesting about direction vectors like **w** is that their norm is equal to 1. That's why we often call them **unit vectors**.

The sum of two vectors

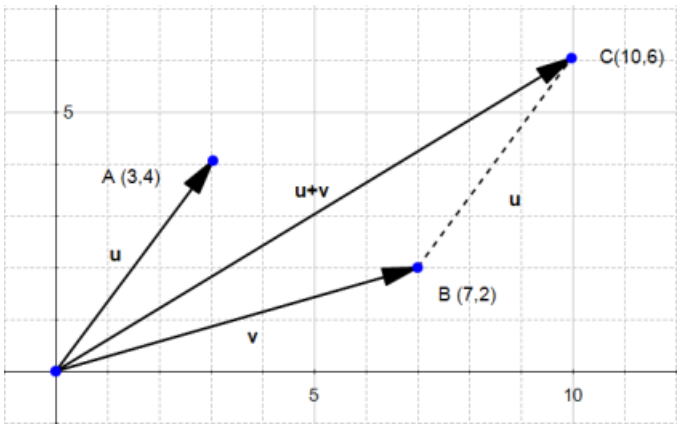


Given two vectors **u**$(u1,u2)$ and **v**$(v1,v2)$

then :

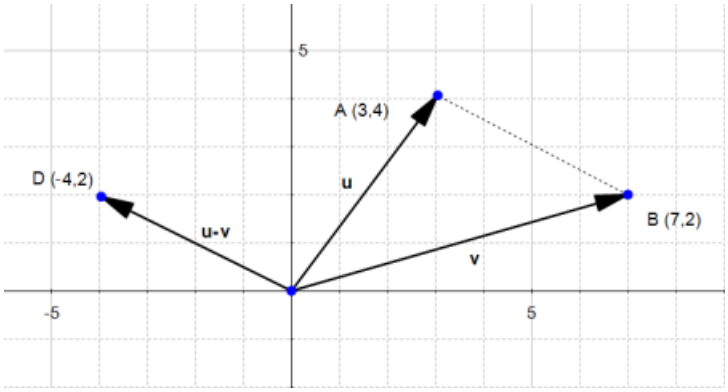
u+**v**$=(u1+v1,u2+v2)$

Which means that adding two vectors gives us **a third vector** whose coordinate are the sum of the coordinates of the original vectors. You can convince yourself with the example below:



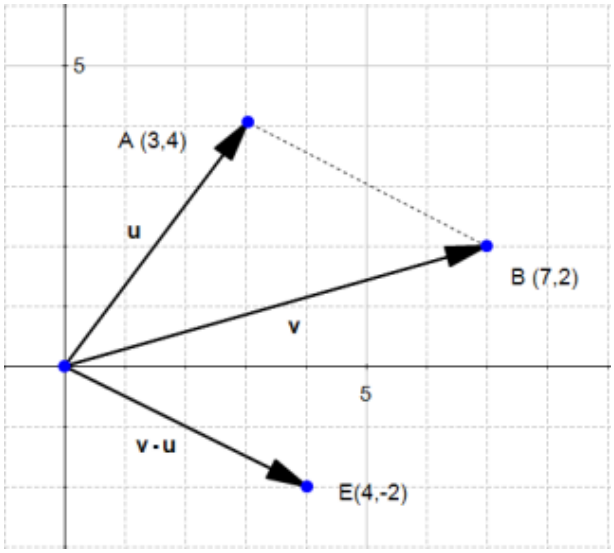
The difference between two vectors

The difference works the same way: $\mathbf{u}-\mathbf{v}=(u1-v1,u2-v2)$



Since the subtraction is not commutative, we can also consider the other case:

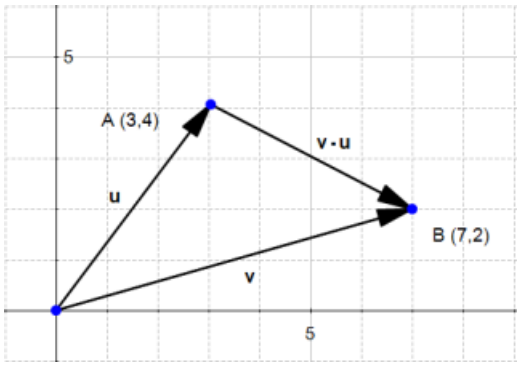
$\mathbf{v}-\mathbf{u}=(v1-u1,v2-u2)$

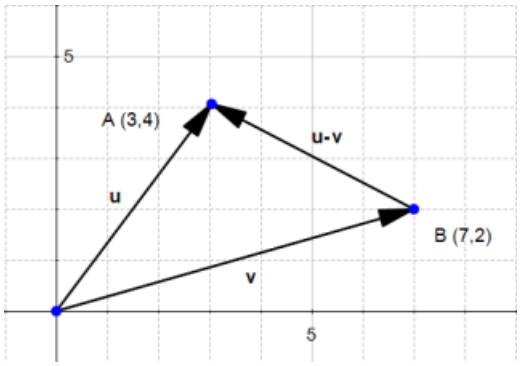


The last two pictures describe the "*true*" vectors generated by the difference of **u** and **v**.

However, since a vector has a magnitude and a direction, we often consider that parallel translate of a given vector (vectors with the same magnitude and direction but with a different origin) are the same vector, just drawn in a different place in space.

So don't be surprised if you meet the following:





If you do the math, it looks wrong, because the end of the vector **u−v** is not in the right point, but it is a convenient way of thinking about vectors which you'll encounter often.

The dot product

One **very** important notion to understand SVM is the dot product.

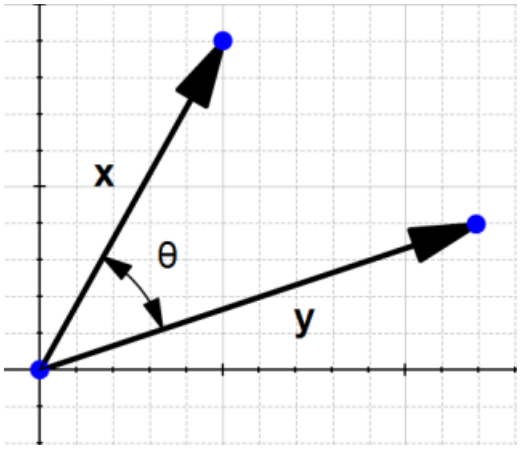
Definition: Geometrically, it is the product of the Euclidian magnitudes of the two vectors and the cosine of the angle between them

Which means if we have two vectors **x** and **y** and there is an angle θ (theta) between them, their dot product is:

$\mathbf{x} \cdot \mathbf{y} = \|x\| \|y\| cos(\theta)$

Why?

To understand let's look at the problem geometrically.



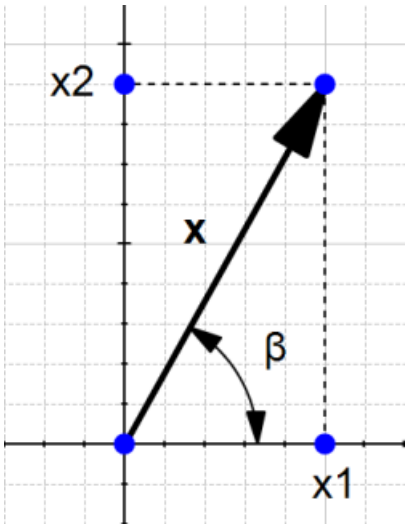
In the definition, they talk about $cos(\theta)$, let's see what it is.

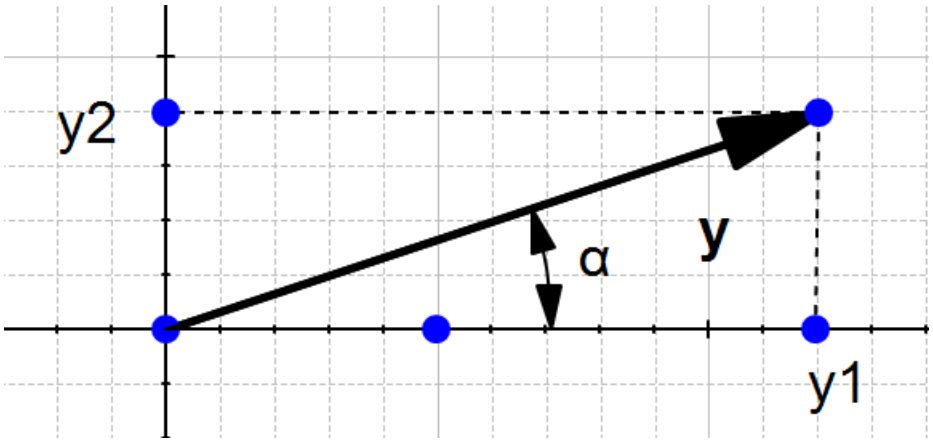
By definition we know that in a right-angled triangle:

$cos(\theta) = adjacent/hypotenuse$

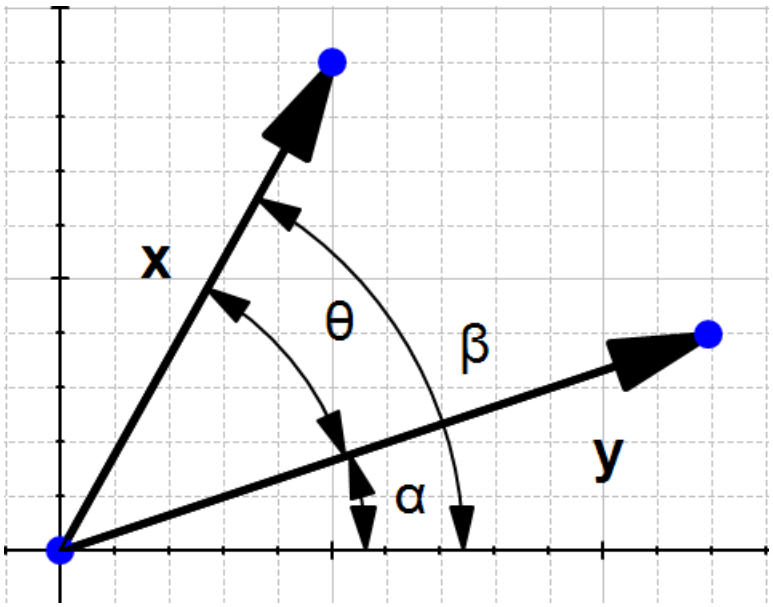
In our example, we don't have a right-angled triangle.

However if we take a different look Figure 12 we can find two right-angled triangles formed by each vector with the horizontal axis.





So now we can view our original schema like this:



We can see that $\theta=\beta-\alpha$

So computing $cos(\theta)$ is like computing $cos(\beta-\alpha)$

There is a special formula called the *difference identity for cosine* which says that:

$cos(\beta-\alpha)=cos(\beta)cos(\alpha)+sin(\beta)sin(\alpha)$

Let's use this formula!

$$cos(\beta) = \frac{adjacent}{hypotenuse} = \frac{x_1}{\|x\|}$$

$$sin(\beta) = \frac{opposite}{hypotenuse} = \frac{x_2}{\|x\|}$$

$$cos(\alpha) = \frac{adjacent}{hypotenuse} = \frac{y_1}{\|y\|}$$

$$sin(\alpha) = \frac{opposite}{hypotenuse} = \frac{y_2}{\|y\|}$$

So if we replace each term

$$cos(\theta) = cos(\beta - \alpha) = cos(\beta)cos(\alpha) + sin(\beta)sin(\alpha)$$

$$cos(\theta) = \frac{x_1}{\|x\|}\frac{y_1}{\|y\|} + \frac{x_2}{\|x\|}\frac{y_2}{\|y\|}$$

$$cos(\theta) = \frac{x_1y_1 + x_2y_2}{\|x\|\|y\|}$$

If we multiply both sides by $\|x\|\|y\|$ we get:

$$\|x\|\|y\|cos(\theta) = x_1y_1 + x_2y_2$$

Which is the same as :

$$\|x\|\|y\|cos(\theta) = \mathbf{x} \cdot \mathbf{y}$$

We just found the geometric definition of the dot product !

Eventually from the two last equations we can see that :

$$\mathbf{x} \cdot \mathbf{y} = x_1y_1 + x_2y_2 = \sum_{i=1}^{2}(x_iy_i)$$

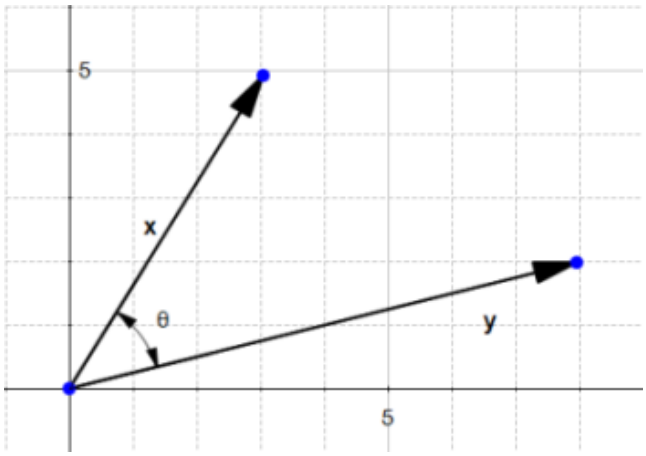
This is the algebraic definition of the dot product !

A few words on notation

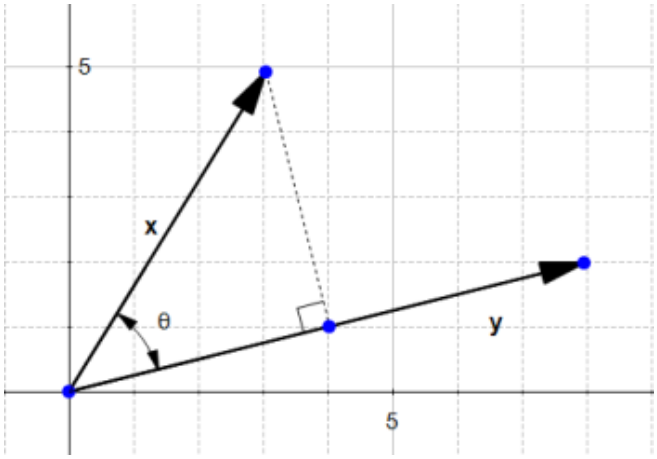
The dot product is called like that because we write a dot between the two vectors. Talking about the dot product **x**·**y** is the same as talking about the **inner product** $\langle x,y \rangle$ (in linear algebra) **scalar product** because we take the product of two vectors and it returns a scalar (a real number)

The orthogonal projection of a vector

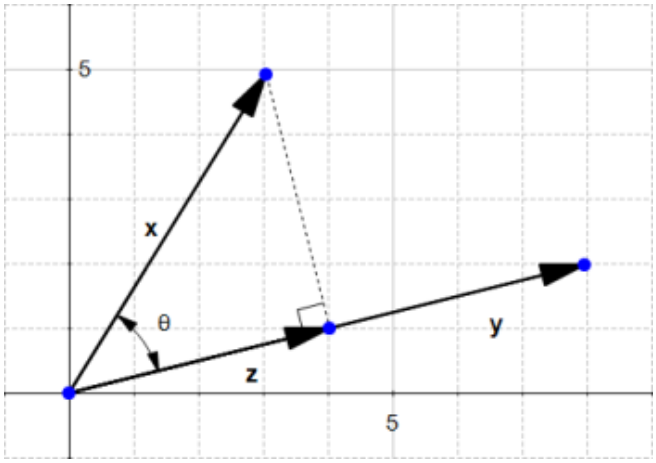
Given two vectors **x** and **y**, we would like to find the orthogonal projection of **x** onto **y**.



To do this we project the vector **x** onto **y**



This gives us the vector **z**



By definition :

$$cos(\theta) = \frac{\|z\|}{\|x\|}$$

$$\|z\| = \|x\| cos(\theta)$$

We saw in the section about the dot product that

$$cos(\theta) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|x\|\|y\|}$$

So we replace $cos(\theta)$ in our equation:

$$\|z\| = \|x\| \frac{\mathbf{x} \cdot \mathbf{y}}{\|x\|\|y\|}$$

$$\|z\| = \frac{\mathbf{x} \cdot \mathbf{y}}{\|y\|}$$

If we define the vector $\mathbf{u}$ as the **direction** of $\mathbf{y}$ then

$$\mathbf{u} = \frac{\mathbf{y}}{\|y\|}$$

and

$$\|z\| = \mathbf{u} \cdot \mathbf{x}$$

We now have a simple way to compute the norm of the vector $\mathbf{z}$.
Since this vector is in the same direction as $\mathbf{y}$ it has the direction $\mathbf{u}$

$$\mathbf{u} = \frac{\mathbf{z}}{\|z\|}$$

$$\mathbf{z} = \|z\|\mathbf{u}$$

And we can say :

> *The vector* $\mathbf{z} = (\mathbf{u} \cdot \mathbf{x})\mathbf{u}$ *is the orthogonal projection of* $\mathbf{x}$ *onto* $\mathbf{y}$.

Why are we interested by the orthogonal projection ? Well in our example, it allows us to compute the distance between $\mathbf{x}$ and the line which goes through $\mathbf{y}$.

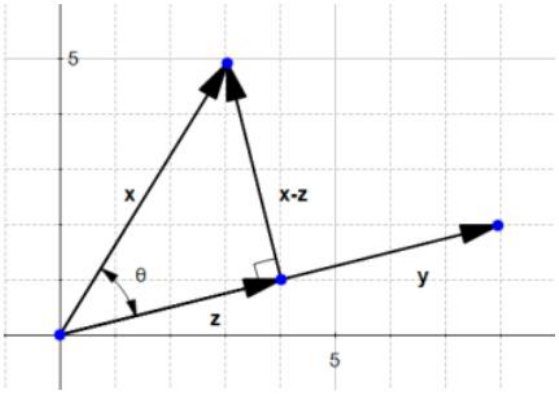


Figure 19

We see that this distance is $\|x - z\|$

$$\|x - z\| = \sqrt{(3-4)^2 + (5-1)^2} = \sqrt{17}$$

The SVM hyperplane: Understanding the equation of the hyperplane

You probably learnt that an equation of a line is: $y=ax+b$. However when reading about hyperplane, you will often find that the equation of a hyperplane is defined by:

$\mathbf{w}^T\mathbf{x}=0$

How does these two forms relate?

In the hyperplane equation you can see that the name of the variables is in bold. Which means that they are vectors? Moreover, $\mathbf{w}^T\mathbf{x}$ is how we compute the inner product of two vectors, and if you recall, the inner product is just another name for the dot product!

Note that $y=ax+b$ is the same thing as $y-ax-b=0$

Given two vectors $\mathbf{w}\begin{pmatrix} -b \\ -a \\ 1 \end{pmatrix}$ and $\mathbf{x}\begin{pmatrix} 1 \\ x \\ y \end{pmatrix}$

$$\mathbf{w}^T\mathbf{x} = -b \times (1) + (-a) \times x + 1 \times y$$

$$\mathbf{w}^T\mathbf{x} = y - ax - b$$

The two equations are just different ways of expressing the same thing.

It is interesting to note that w_0 is $-b$, which means that this value determines the intersection of the line with the vertical axis.

Why do we use the hyperplane equation $\mathbf{w}^T\mathbf{x}$ instead of $y = ax + b$?

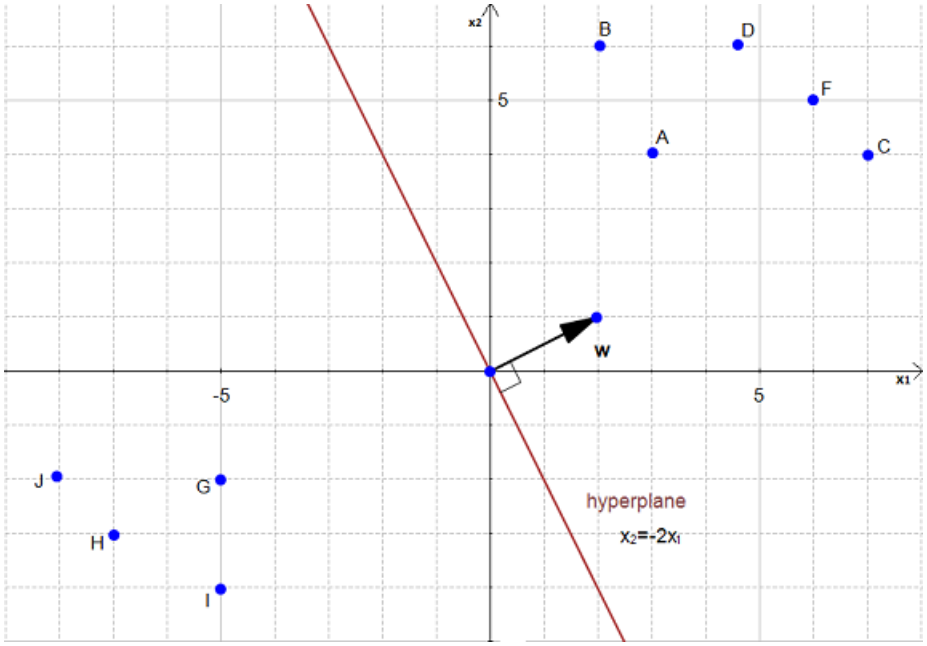
For two reasons:

- it is easier to work in more than two dimensions with this notation,
- the vector **w** will always be normal to the hyperplane(Note: I received a lot of questions about the last remark. **W** will always be normal because we use this vector to define the hyperplane, so by definition it will be normal. As you can see this page, when we define a hyperplane, we suppose that we have a vector that is orthogonal to the hyperplane)

And this last property will come in handy to compute the distance from a point to the hyperplane.

Compute the distance from a point to the hyperplane

In Figure 20 we have a hyperplane, which separates two groups of data.



To simplify this example, we have set $w0=0$.

As you can see on the Figure 20, the equation of the hyperplane is:

$x2=-2x1$ which is equivalent to

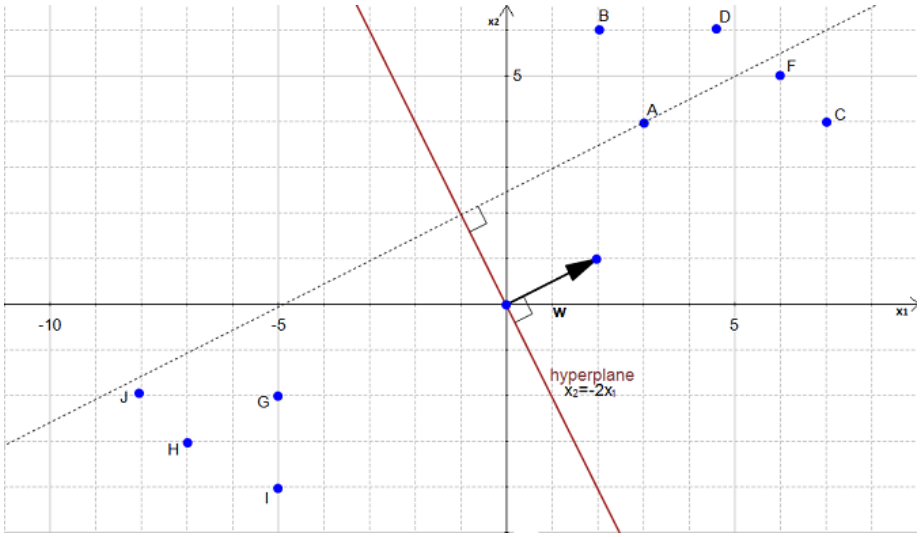
$\mathbf{w}^T\mathbf{x}=0$

with $\mathbf{w}\begin{pmatrix}2\\1\end{pmatrix}$ and $\mathbf{x}\begin{pmatrix}x_1\\x_2\end{pmatrix}$

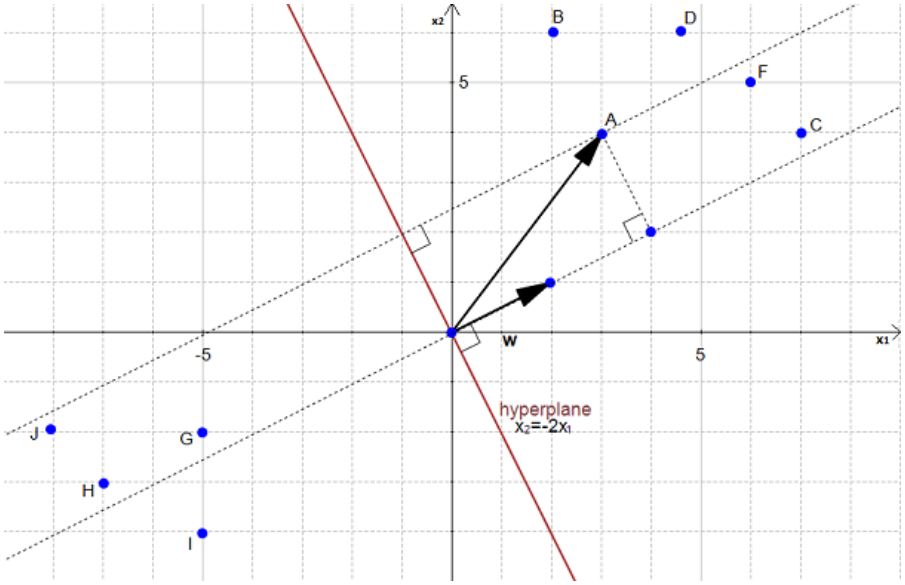
Note that the vector $\mathbf{w}$ is shown on the Figure 20. (w is not a data point)

We would like to compute the distance between the point $A(3,4)$ and the hyperplane.

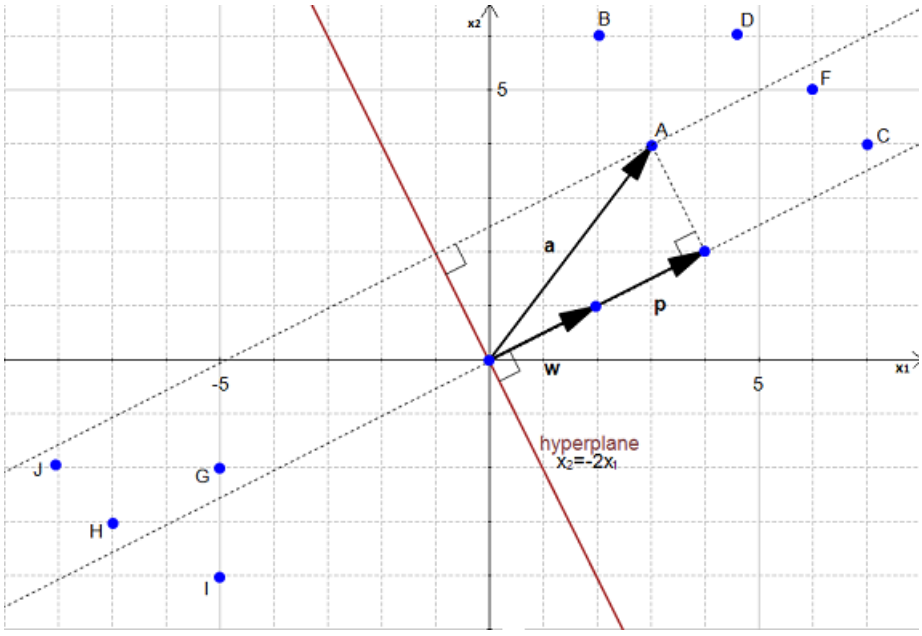
This is the distance between A and its projection onto the hyperplane



We can view point *A* as a vector from origin to *A*. If we project it onto the normal vector **w**



We get the vector **p**



Our goal is to find the distance between the point $A(3,4)$ and the hyperplane.
We can see in Figure 23 that this distance is the same thing as $\|p\|$.
Let's compute this value.

We start with two vectors, $\mathbf{w} = (2,1)$ which is normal to the hyperplane, and $\mathbf{a} = (3,4)$ which is the vector between the origin and A.

$$\|w\| = \sqrt{2^2 + 1^2} = \sqrt{5}$$

Let the vector $\mathbf{u}$ be the direction of $\mathbf{w}$

$$\mathbf{u} = (\frac{2}{\sqrt{5}}, \frac{1}{\sqrt{5}})$$

$\mathbf{p}$ is the orthogonal projection of $\mathbf{a}$ onto $\mathbf{w}$ so :

$$\mathbf{p} = (\mathbf{u} \cdot \mathbf{a})\mathbf{u}$$

$$\mathbf{p} = (3 \times \frac{2}{\sqrt{5}} + 4 \times \frac{1}{\sqrt{5}})\mathbf{u}$$

$$\mathbf{p} = (\frac{6}{\sqrt{5}} + \frac{4}{\sqrt{5}})\mathbf{u}$$

$$\mathbf{p} = \frac{10}{\sqrt{5}}\mathbf{u}$$

$$\mathbf{p} = (\frac{10}{\sqrt{5}} \times \frac{2}{\sqrt{5}}, \frac{10}{\sqrt{5}} \times \frac{1}{\sqrt{5}})$$

$$\mathbf{p} = (\frac{20}{5}, \frac{10}{5})$$

$$\mathbf{p} = (4,2)$$

$$\|p\| = \sqrt{4^2 + 2^2} = 2\sqrt{5}$$

Compute the margin of the hyperplane

Now that we have the distance $\|p\|$ between A and the hyperplane, the margin is defined by :

$$margin = 2\|p\| = 4\sqrt{5}$$

We did it ! We computed the margin of the hyperplane !

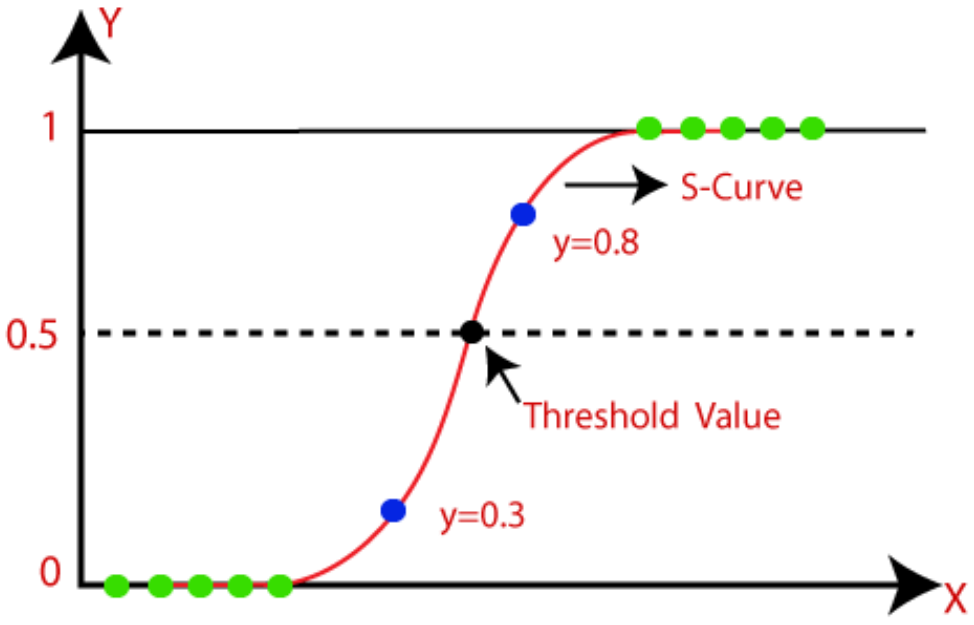
Topic: 34. Logistic Regression

Theory	Mathematics	Numerical
✓	✓	✗

Theory questions

34. Why use Logistic Regression?

- Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.
- Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, **it gives the probabilistic values which lie between 0 and 1**.
- Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas **Logistic regression is used for solving the classification problems**.
- In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.
- Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets. Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification. The below image is showing the logistic function:



Note: Logistic regression uses the concept of predictive modeling as regression; therefore, it is called logistic regression, but is used to classify samples; Therefore, it falls under the classification algorithm.

35. Explain principle of Logistic Regression.

Logistic Function (Sigmoid Function):

- The sigmoid function is a mathematical function used to map the predicted values to probabilities.
- It maps any real value into another value within a range of 0 and 1.
- The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form. The S-form curve is called the sigmoid function or the logistic function.
- In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

Assumptions for Logistic Regression:

- The dependent variable must be categorical in nature.
- The independent variable should not have multi-collinearity.

Logistic Regression Equation:

The Logistic regression equation can be obtained from the Linear Regression equation. The mathematical steps to get Logistic Regression equations are given below:

We know the equation of the straight line can be written as:

$$y= b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

In Logistic Regression y can be between 0 and 1 only, so for this let's divide the above equation by (1-y):

$$\frac{y}{1-y}$$; 0 for y= 0, and infinity for y=1

But we need range between -[infinity] to +[infinity], then take logarithm of the equation it will become:

$$log\left[\frac{y}{1-y}\right] = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

36. State types of Logistic Regression?

On the basis of the categories, Logistic Regression can be classified into three types:

- **Binomial:** In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc.

- **Multinomial:** In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep"
- **Ordinal:** In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as "low", "Medium", or "High".

37. What are advantages and limitations of the Logistic Regression?

Advantages	**Disadvantages**
Logistic regression is easier to implement, interpret, and very efficient to train.	If the number of observations is lesser than the number of features, Logistic Regression should not be used; otherwise, it may lead to overfitting.
It makes no assumptions about distributions of classes in feature space.	It constructs linear boundaries.
It can easily extend to multiple classes (multinomial regression) and a natural probabilistic view of class predictions.	Limitation of Logistic Regression is the assumption of linearity between dependent variable and independent variables.
It not only provides a measure of how appropriate a predictor (coefficient size)is, but also its direction of association (positive or negative).	It can only be used to predict discrete functions. Hence, the dependent variable of Logistic Regression is bound to the discrete number set.
It is very fast at classifying unknown records.	Non-linear problems can't be solved with logistic regression because it has a linear decision surface. Linearly separable data is rarely found in real-world scenarios.
Good accuracy for many simple data sets and it performs well when the dataset is linearly separable.	Logistic Regression requires average or no multicollinearity between independent variables.
It can interpret model coefficients as indicators of feature importance.	It is tough to obtain complex relationships using logistic regression. More powerful and compact algorithms such as Neural Networks can easily outperform this algorithm.
Logistic regression is less inclined to over-fitting but it can overfit in high dimensional datasets.One may consider Regularization (L1 and L2) techniques to avoid over-fittingin these scenarios.	In Linear Regression independent and dependent variables are related linearly. But Logistic Regression needs that independent variables are linearly related to the log odds (log(p/(1-p)).

38. Differentiate between logistic regression and linear regression?

Linear Regression	Logistic Regression
Linear regression is used to predict the continuous dependent variable using a given set of independent variables.	Logistic Regression is used to predict the categorical dependent variable using a given set of independent variables.
Linear Regression is used for solving Regression problem.	Logistic regression is used for solving Classification problems.
In Linear regression, we predict the value of continuous variables.	In logistic Regression, we predict the values of categorical variables.
In linear regression, we find the best fit line, by which we can easily predict the output.	In Logistic Regression, we find the S-curve by which we can classify the samples.
Least square estimation method is used for estimation of accuracy.	Maximum likelihood estimation method is used for estimation of accuracy.
The output for Linear Regression must be a continuous value, such as price, age, etc.	The output of Logistic Regression must be a Categorical value such as 0 or 1, Yes or No, etc.
In Linear regression, it is required that relationship between dependent variable and independent variable must be linear.	In Logistic regression, it is not required to have the linear relationship between the dependent and independent variable.
In linear regression, there may be collinearity between the independent variables.	In logistic regression, there should not be collinearity between the independent variable.
Linear Regression is a supervised regression model.	Logistic Regression is a supervised classification model.
In Linear Regression, we predict the value by an integer number.	In Logistic Regression, we predict the value by 1 or 0.
Here no activation function is used.	Here activation function is used to convert a linear regression equation to the logistic regression equation
Here no threshold value is needed.	Here a threshold value is added.
Here we calculate Root Mean Square Error(RMSE) to predict the next weight value.	Here we use precision to predict the next weight value.

Here dependent variable should be numeric and the response variable is continuous to value.	Here the dependent variable consists of only two categories. Logistic regression estimates the odds outcome of the dependent variable given a set of quantitative or categorical independent variables.
It is based on the least square estimation.	It is based on maximum likelihood estimation.
Here when we plot the training datasets, a straight line can be drawn that touches maximum plots.	Any change in the coefficient leads to a change in both the direction and the steepness of the logistic function. It means positive slopes result in an S-shaped curve and negative slopes result in a Z-shaped curve.
Linear regression is used to estimate the dependent variable in case of a change in independent variables. For example, predict the price of houses.	Whereas logistic regression is used to calculate the probability of an event. For example, classify if tissue is benign or malignant.
Linear regression assumes the normal or gaussian distribution of the dependent variable.	Logistic regression assumes the binomial distribution of the dependent variable.

The Similarities between Linear Regression and Logistic Regression

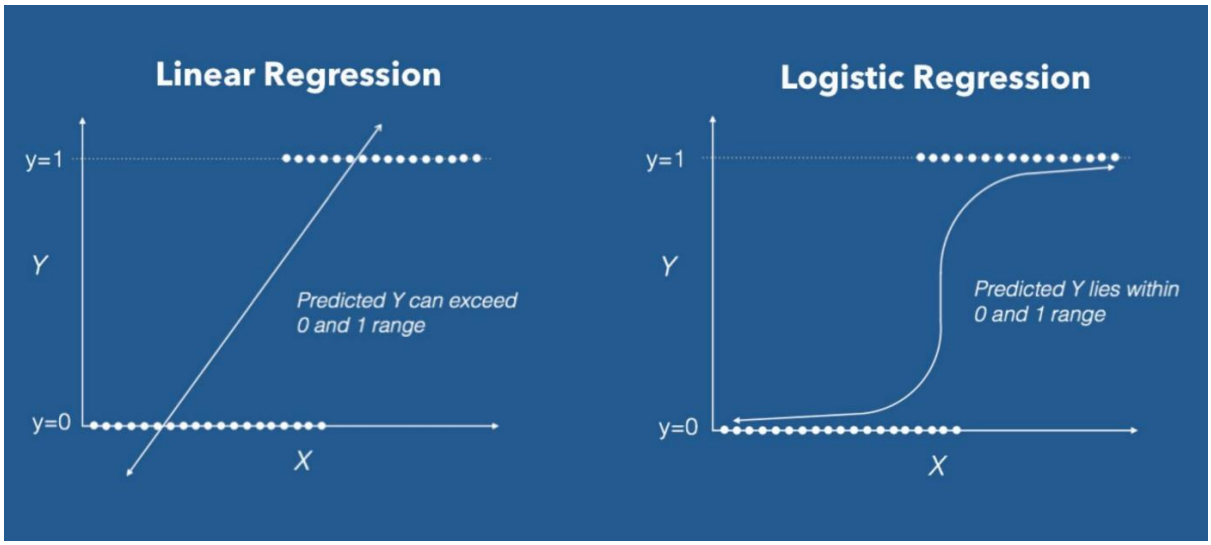
- Linear Regression and Logistic Regression both are supervised Machine Learning algorithms.
- Linear Regression and Logistic Regression, both the models are parametric regression i.e. both the models use linear equations for predictions

That's all the similarities we have between these two models.

However, functionality-wise these two are completely different. Following are the differences.

The Differences between Linear Regression and Logistic Regression

- Linear Regression is used to handle regression problems whereas Logistic regression is used to handle the classification problems.
- Linear regression provides a continuous output but Logistic regression provides discreet output.
- The purpose of Linear Regression is to find the best-fitted line while Logistic regression is one step ahead and fitting the line values to the sigmoid curve.
- The method for calculating loss function in linear regression is the mean squared error whereas for logistic regression it is maximum likelihood estimation.



Mathematics based questions

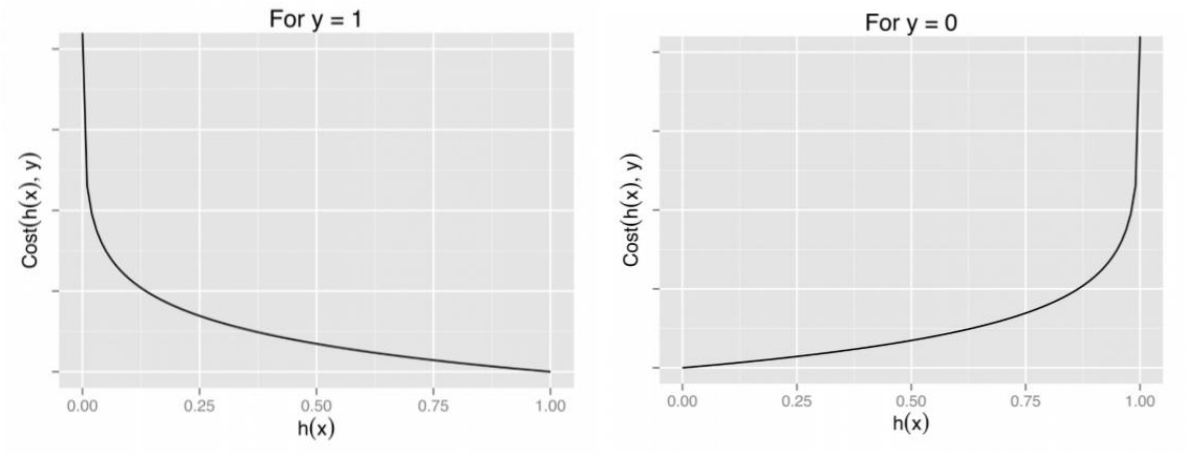
39. Cost function

The Cost function of linear regression

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2.$$

Cost function of Logistic Regression

$$Cost(h_\theta(x), y) = \begin{cases} -log(h_\theta(x)) & \text{if } y = 1 \\ -log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$



The *i* indexes have been removed for clarity. In words this is the cost the algorithm pays if it predicts a value *hθ*(*x*) while the actual cost label turns out to be *y*. By using this function we will grant the *convexity* to the function the gradient descent algorithm has to process, as discussed above. There is also a mathematical proof for that, which is outside the scope of this introductory course. In case *y*=1, the output (i.e. the cost to pay) approaches to 0 as

$h\theta(x)$ approaches to 1. Conversely, the cost to pay grows to infinity as $h\theta(x)$ approaches to 0. This is a desirable property: we want a bigger penalty as the algorithm predicts something far away from the actual value. If the label is y=1 but the algorithm predicts $h\theta(x)$=0, the outcome is completely wrong. Conversely, the same intuition applies when y=0, depicted in the plot 2. below, right side. Bigger penalties when the label is y=0 but the algorithm predicts $h\theta(x)$=1. The above two functions can be compressed into a single function i.e.

$$J(\theta) = -\frac{1}{m}\sum\left[y^{(i)}\log(h\theta(x(i))) + \left(1-y^{(i)}\right)\log(1-h\theta(x(i)))\right]$$

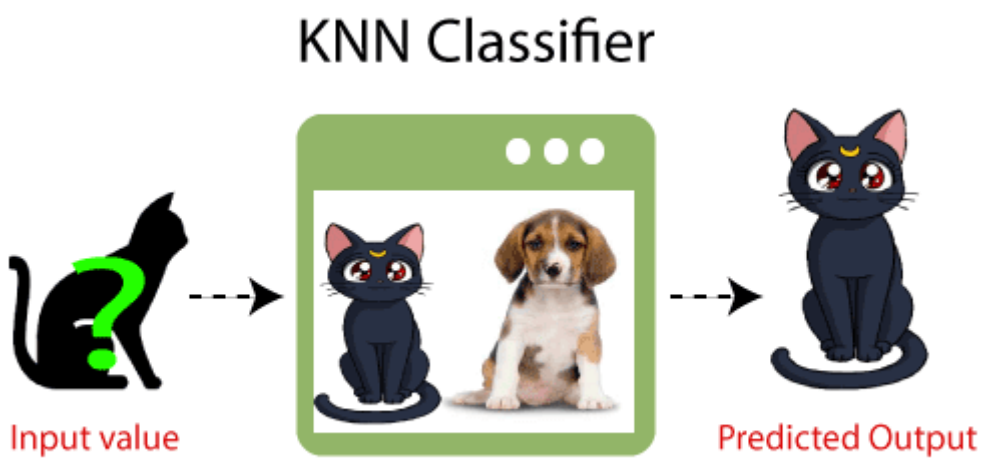
Topic: K-Means & K-Nearest Neighbor (KNN)

Theory	Mathematics	Numerical
✓	✓	✗

Theory questions

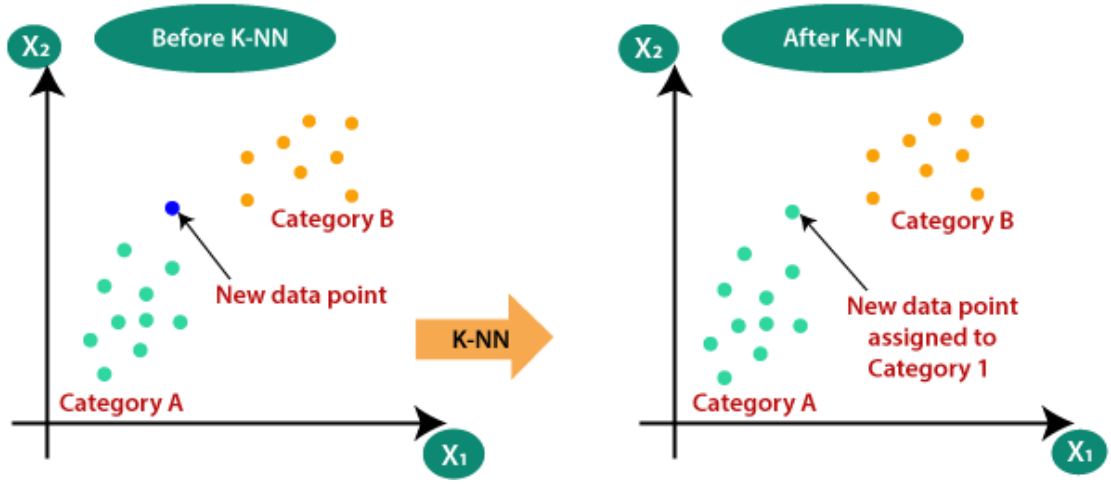
40. What is meant by K Nearest Neighbor algorithm?

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- 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 algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- 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 algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
- **Example:** Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.



41. Why do we need a K-NN Algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:

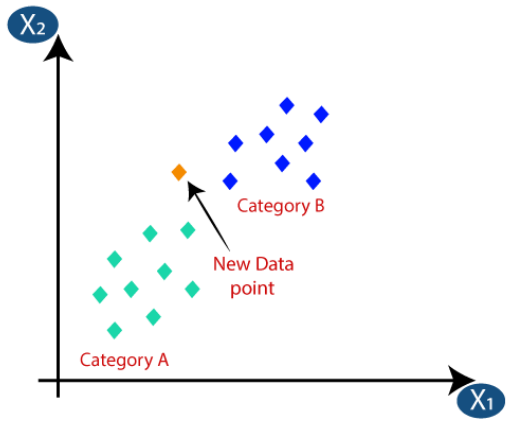


42. How does K-NN work?

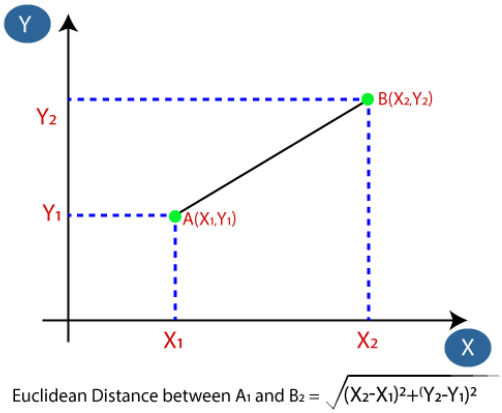
The K-NN working can be explained on the basis of the below algorithm:

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

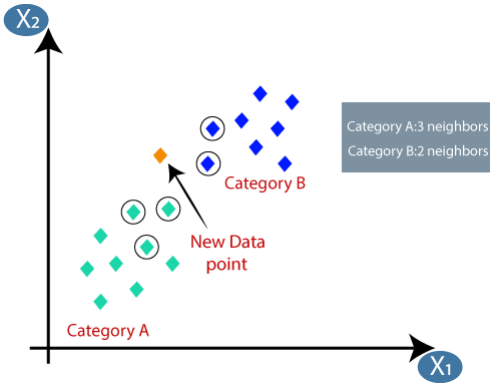
Suppose we have a new data point and we need to put it in the required category. Consider the below image:



- Firstly, we will choose the number of neighbors, so we will choose the k=5.
- Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



- By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:



- As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

43. What is the difference between KNN and K means?

- K-NN is a **Supervised** while K-means is an **unsupervised** Learning.
- K-NN is a **classification** or **regression** machine learning algorithm while K-means is a **clustering** machine learning algorithm.
- K-NN is a **lazy learner** while K-Means is an **eager learner**. An eager learner has a model fitting that means a training step but a lazy learner does not have a training phase.
- K-NN performs much better if all of the data have the same scale but this is not true for K-means.
- K-means is a clustering algorithm that tries to partition a set of points into K sets (clusters) such that the points in each cluster tend to be near each other. It is unsupervised because the points have no external classification.
- K-nearest neighbors is a classification (or regression) algorithm that in order to determine the classification of a point, combines the classification of the K nearest points. It is supervised because you are trying to classify a point based on the known classification of other points.

44. What is K-means used for?

K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science. In this topic, we will learn what is K-means clustering algorithm, how the algorithm works, along with the Python implementation of k-means clustering.

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.

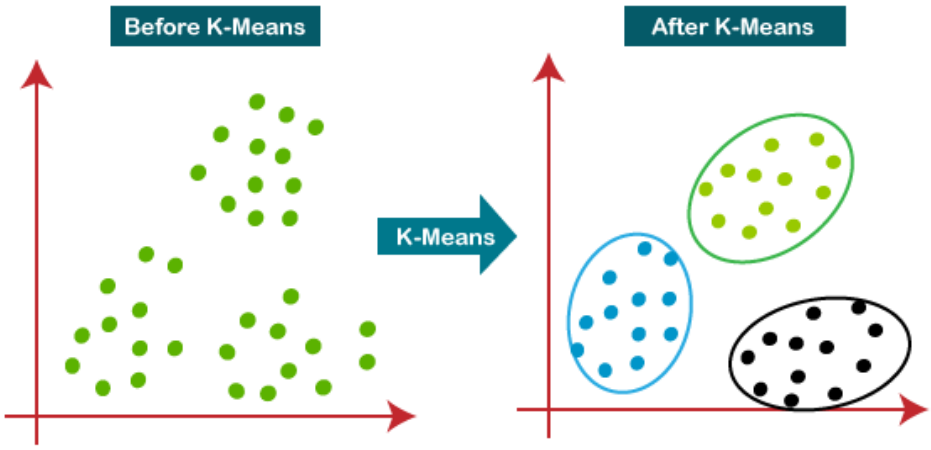
It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

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:

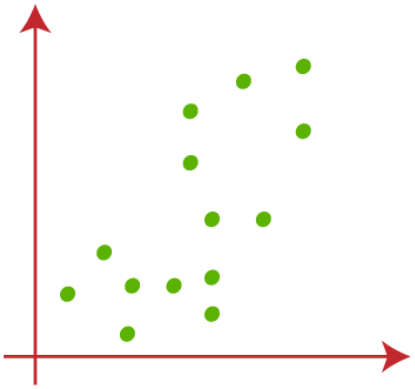


45. How does K-means work?

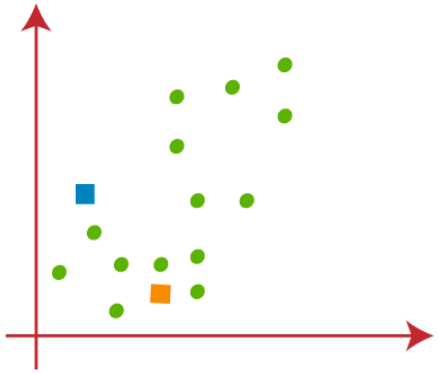
The working of the K-Means algorithm is explained in the below steps:

- **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.

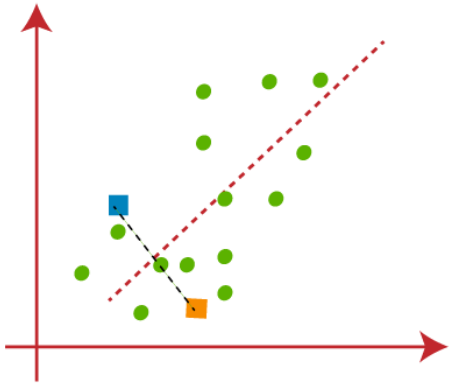
Suppose we have two variables M1 and M2. The x-y axis scatter plot of these two variables is given below:



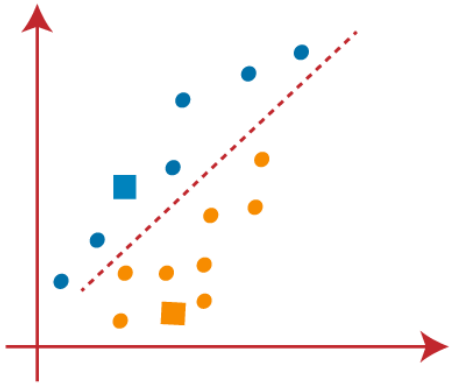
- Let's take number k of clusters, K=2, to identify dataset and to put them into different clusters. It means here we will try to group these datasets into 2 different clusters.
- We need to choose some random k points or centroid to form the cluster. These points can be either the points from the dataset 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:



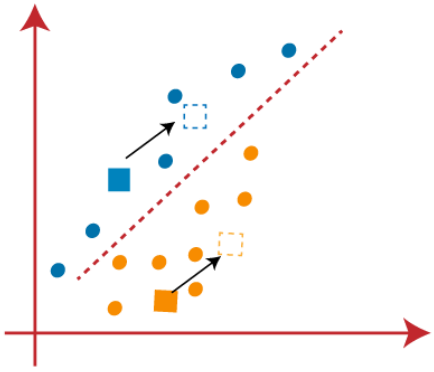
- Now we will assign each data point of the scatter plot to its closest K-point or centroid. We will compute it by applying some mathematics that we have studied to calculate the distance between two points. So, we will draw a median between both the centroids. Consider the below image:



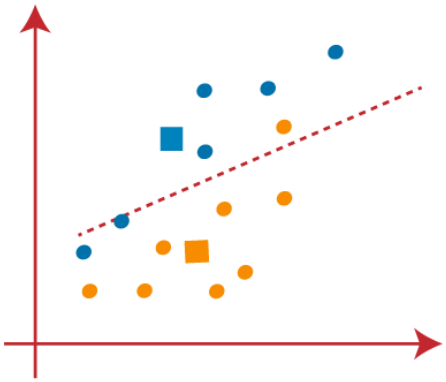
- From the above image, it is clear that points left side of the line is near to the K1 or blue centroid, and points to the right of the line are close to the yellow centroid. Let's color them as blue and yellow for clear visualization.



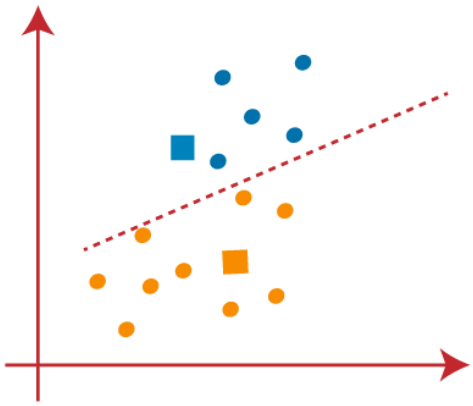
- As we need to find the closest cluster, so we will repeat the process by choosing **a new centroid**. To choose the new centroids, we will compute the center of gravity of these centroids, and will find new centroids as below:



- 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:\

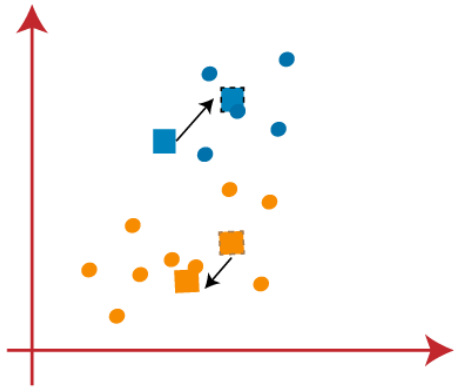


From the above image, we can see, one yellow point is on the left side of the line, and two blue points are right to the line. So, these three points will be assigned to new centroids.

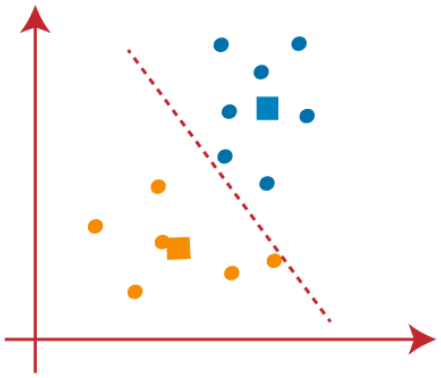


As reassignment has taken place, so we will again go to the step-4, which is finding new centroids or K-points.

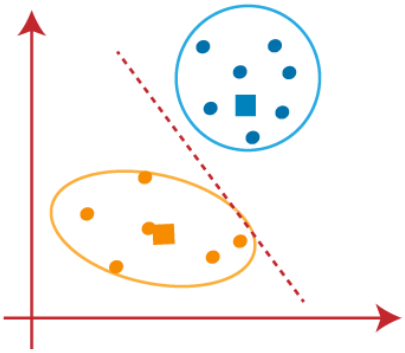
- We will repeat the process by finding the center of gravity of centroids, so the new centroids will be as shown in the below image:



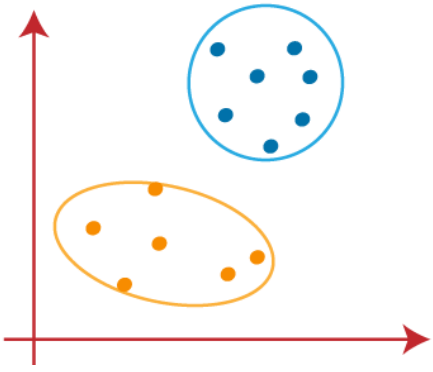
- As we got the new centroids so again will draw the median line and reassign the data points. So, the image will be:



- 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:



46. Is K nearest neighbor supervised or unsupervised?

KNN is a simple supervised learning algorithm.

KNN works on a basic assumption that data points of similar classes are closer to each other.

Now suppose you have a classification problem to identify whether a given data point is of class A or class B and your aim is to classify test datapoints into these classes for which you have a training dataset of alreday classified data points.

KNN assignes 1/k probability to 'k' nearest pre classified training data point from our test data point and 0 probability to rest of data points, where k may be any number (but try to put it odd to avoid tie cases). After that we count the number of each classes (i.e A and B) out of those K points and our test data point will be classified as that class whose count is greater.

For example if we are using k=5 then our algorithm looks for 5 nearest point from our test dataset point and count the number of each class out of those 5 point.Suppose our counting results in A=3 and B=2 then KNN will assign class A to that test dataset point.

KNN is categorized under supervised ML techniques. It works assuming that similar classes data points are near one another. Take a case of data points classification, where they fit? Class A or B? For this you have a classified data points training set. KNN assumes a probability of 1/k for 'k' closest data point, then assumes zero probability for other data points.

Now k could be any digit, let's count the classes (A, B) out of the k points, the test data points get classified as the class with higher count. If k = 6, then the algorithm searches for the nearest point in the dataset, and count instances of every class out of the 6 point. Now assume counting turns out to be A = 2, B = 1, then KNN will allot A to the data set point.

In this scenario, Data points classification is done considering their proximity with known Data points classes, thus KNN is a supervised ML algorithm.

47. What are advantages and limitations of KNN and K means?

The k-nearest neighbors (KNN) algorithm is a simple, supervised machine learning algorithm that can be used to solve both classification and regression problems. It's easy to implement and understand, but has a major drawback of becoming significantly slows as the size of that data in use grows. KNN works by finding the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query, then votes for the most frequent label (in the case of classification) or averages the labels (in the case of regression). In the case of classification and regression, we saw that choosing the right K for our data is done by trying several Ks and picking the one that works best.

Advantages

- The algorithm is simple and easy to implement.
- There's no need to build a model, tune several parameters, or make additional assumptions.
- The algorithm is versatile. It can be used for classification, regression, and search (as we will see in the next section).

Disadvantages

- The algorithm gets significantly slower as the number of examples and/or predictors/independent variables increase.

Mathematics based questions

48. How to choose right value for K in KNN?

To select the K that's right for your data, we run the KNN algorithm several times with different values of K and choose the K that reduces the number of errors we encounter while maintaining the algorithm's ability to accurately make predictions when it's given data it hasn't seen before. Here are some things to keep in mind:

- As we decrease the value of K to 1, our predictions become less stable. Just think for a minute, imagine K=1 and we have a query point surrounded by several reds and one green (I'm thinking about the top left corner of the colored plot above), but the green is the single nearest neighbor. Reasonably, we would think the query point is most likely red, but because K=1, KNN incorrectly predicts that the query point is green.
- Inversely, as we increase the value of K, our predictions become more stable due to majority voting / averaging, and thus, more likely to make more accurate predictions (up to a certain point). Eventually, we begin to witness an increasing number of errors. It is at this point we know we have pushed the value of K too far.
- In cases where we are taking a majority vote (e.g. picking the mode in a classification problem) among labels, we usually make K an odd number to have a tiebreaker.

49. How to choose the value of "K number of clusters" in K-means Clustering?

The performance of the K-means clustering algorithm depends upon highly efficient clusters that it forms. But choosing the optimal number of clusters is a big task. There are some different ways to find the optimal number of clusters, but here we are discussing the most appropriate method to find the number of clusters or value of K. The method is given below:

Elbow Method

The Elbow method is one of the most popular ways to find the optimal number of clusters. This method uses the concept of WCSS value. **WCSS** stands for **Within Cluster Sum of**

Squares, which defines the total variations within a cluster. The formula to calculate the value of WCSS (for 3 clusters) is given below:

$WCSS = \sum_{Pi \text{ in Cluster1}} distance(P_i\ C_1)^2 + \sum_{Pi \text{ in Cluster2}} distance(P_i\ C_2)^2 + \sum_{Pi \text{ in CLuster3}} distance(P_i\ C_3)^2$

In the above formula of WCSS,

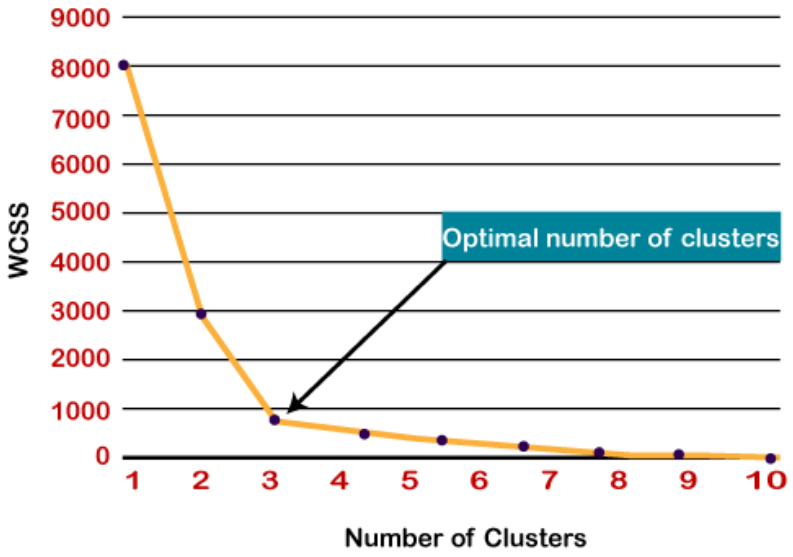
$\sum_{Pi \text{ in Cluster1}} distance(P_i\ C_1)^2$: It is the sum of the square of the distances between each data point and its centroid within a cluster1 and the same for the other two terms.

To measure the distance between data points and centroid, we can use any method such as Euclidean distance or Manhattan distance.

To find the optimal value of clusters, the elbow method follows the below steps:

- It executes the K-means clustering on a given dataset for different K values (ranges from 1-10).
- For each value of K, calculates the WCSS value.
- Plots a curve between calculated WCSS values and the number of clusters K.
- The sharp point of bend or a point of the plot looks like an arm, then that point is considered as the best value of K.

Since the graph shows the sharp bend, which looks like an elbow, hence it is known as the elbow method. The graph for the elbow method looks like the below image:



Note: We can choose the number of clusters equal to the given data points. If we choose the number of clusters equal to the data points, then the value of WCSS becomes zero, and that will be the endpoint of the plot.

Feel free to contact me on +91-8329347107 calling / +91-9922369797 whatsapp, email ID: adp.mech@coep.ac.in and abhipatange93@gmail.com

****